Universidad Pública de Navarra

ESCUELA TECNICA SUPERIOR
DE INGENIEROS AGRONOMOS

Nafarroako Unibertsitate Publikoa

NEKAZARITZAKO INGENIARIEN
GOI MAILAKO ESKOLA TEKNIKOA

GESTIÓN Y VISUALIZACIÓN DE DATOS GEOGRÁFICOS AFECTADOS EN PLANES DE EMERGENCIAS E INUNDACIONES

presentado por

BEÑAT USKOLA IBARRA *(e)k*

aurkeztua

MÁSTER EN SISTEMAS DE INFORMACIÓN GEOGRÁFICA Y TELEDETECCIÓN
MASTERRA INFORMAZIO SISTEMA GEOGRAFIKOETAN ETA TELEDETEKZIOAN

Septiembre, 2016 / *2016ko iraila*

Resumen

La aplicación web NOE de Tesicnor es un 'Sistema Informático de Ayuda en la Gestión de Emergencias por Inundaciones' que pretende convertir el 'Plan de Emergencias para Inundaciones' de una localidad en una aplicación web fiable, útil y sencilla. Dentro de esta aplicación existe un visor GIS donde se pretende visualizar las zonas afectadas en cada nivel de emergencia así como las acciones a realizar, facilitándole al responsable la gestión de la situación.

El objetivo es diseñar y desarrollar un sistema de carga de datos geolocalizados afectados por las inundaciones y su inclusión en herramientas GIS libres que puedan ser integrables desde la aplicación web. Para poder llevarlo a cabo se ha construido una arquitectura web GIS integral y se han creado procedimientos para facilitar el proceso desde la descarga de los datos ya existentes hasta la representación en el propio visor de los datos de interés.

Palabras clave:

Web Mapping, Inundaciones, Cartografía, SIG, IDE, PostGIS, GeoServer, OpenLayers

Abstract

Tesicnor's web application NOE is a 'Computer System that helps in the Management of Flood Emergencies' which aims to make the 'Emergency Flood Plan' of a town in a reliable, useful and simple web application. Within this application there is a GIS viewer which aims to visualize the affected areas at each level of emergency and also the actions to take, thereby facilitating to the responsible the management of the situation.

The goal is to design and develop a charging system of geolocated data affected by the floods and their inclusion in free GIS tools that can be integrated from the Web application. In order to carry it out it has built a comprehensive GIS web architecture and have created procedures to facilitate all the process.

Keywords:

Web Mapping, Flood, Cartography, GIS, SDI, PostGIS, GeoServer, OpenLayers

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN: ANTECEDENTES Y OBJETIVOS

1.1. INUNDACIONES

Las inundaciones son la catástrofe natural que mayores daños genera en España. Según el Consorcio de Compensación de Seguros y el Instituto Geológico y Minero de España, los daños por inundaciones se estiman en total en una media de 800 millones de euros anuales. A modo de ejemplo, cabe destacar que sólo en bienes asegurados, en el período 1971-2012, según las estadísticas del Consorcio, el 42,9% de los expedientes tramitados han sido debidos a daños por inundaciones, que han supuesto el 60,3% del total de las indemnizaciones, las cuales, de media, suponen más de 130 millones de euros cada año (MAGRAMA, 2016). Sin olvidar que puede peligrar la vida e integridad física de las personas. Como ejemplo la inundación ocurrida en Pamplona en junio de 2013 de la figura 1.

Figura 1. Imagen aérea de la inundación de junio de 2013 en Pamplona/Iruña (fuente: pamplonaactual.com)

Las competencias en gestión y defensa frente a los efectos adversos de las inundaciones afectan a todas las administraciones, desde la local en las labores de planeamiento urbanístico y protección civil, la autonómica, en material de ordenación del territorio, protección civil y gestión del dominio público hidráulico en las cuencas intracomunitarias y la estatal, en relación con protección civil, la gestión del dominio público hidráulico en las cuencas intercomunitarias y la gestión del dominio público marítimo terrestre en las inundaciones causadas en las zonas de transición y las debidas a la elevación del nivel del mar.

Como refuerzo a todas estas actuaciones, la Comisión Europea aprobó en noviembre de 2007 la Directiva 2007/60 sobre la evaluación y gestión de las inundaciones (EU, 2007) que ha sido transpuesta a la legislación española mediante el Real Decreto 903/2010 de evaluación y gestión de riesgos de inundación (Boletín oficial del Estado, 2010). La implantación de esta Directiva supone una oportunidad para mejorar la coordinación de todas las administraciones a la hora de reducir estos daños, centrándose fundamentalmente en las zonas con mayor riesgo de inundación, llamadas Áreas de Riesgo Potencial Significativo de Inundación (ARPSIs).

Por desarrollo de esta normativa Europea y Nacional será obligatorio la redacción de 'Planes de Actuación Municipal' de varios municipios en la Comunidad Foral, cuyo elaboración y aprobación corresponde al órgano de Gobierno Municipal, y la homologación se realizará por el Departamento de Protección Civil de Navarra (Gastesi, 2016). Estos planes de actuación podrán ser tanto para Emergencias ante Inundaciones como Emergencias de Presas.

El objetivo básico del Plan es que la organización municipal y la población, se guíen por un dispositivo permanente y actualizado de información, previsión, alerta y actuación ante estas emergencias con capacidad de proteger a la población amenazada y, en lo posible, evitar o al menos reducir los daños que puedan producir a los bienes y servicios esenciales, de acuerdo con los medios y recursos locales disponibles.

Estos planes recogen los procedimientos de actuación ante una inundación, un catálogo de los elementos vulnerables y de las zonas del municipio en función del riesgo que tienen, y una relación actualizada de medios y recursos locales que puedan ser utilizados en caso de emergencia. Se pone especial énfasis en la autoprotección.

El documento que se crea para describir esta planificación asigna ciertos umbrales pluvio-hidrológicos del río que marcan el nivel de emergencia, por ejemplo en el Plan del Baztan se establece: a 126 m^3/s le corresponde la 'Pre-emergencia', a 181 m^3/s 'Emergencia 0', a 181 m^3/s pero con mayor intensidad 'Emergencia 1', a 267 m^3/s 'Emergencia 2' y a 320 m^3/s 'Emergencia 3'.

Además, para cada nivel de emergencia se crea una ficha de respuesta donde se definen los elementos afectados, las medidas de seguridad o acciones a adoptar (permanentes, temporales y graduales para las diferentes tipologías de activos a proteger) y los responsables de su implantación, plazos, mecanismos de coordinación y seguimiento. En los anexos se describen:

- Viviendas con peligro de quedar aislados: depende del periodo de retorno de los caudales circundantes.
- Carreteras cortadas. Vías de Evacuación.
- Medios de aplicación de alerta a la población.
- Catálogo de medios, recursos y teléfonos de interés.
- Instrucciones a la población.
- Seguimiento Pluvio-hidrológico.

1.2. TESICNOR Y LA APLICACIÓN WEB NOE

En este contexto se sitúa la aplicación web NOE desarrollada por la empresa Tesicnor (figura 2). Tesicnor es una consultoría técnica especializada en garantizar la seguridad en el ámbito laboral, industrial y ciudadana, así como en la optimización de los recursos energéticos. Trabaja para ofrecer a sus clientes y a la sociedad soluciones prácticas, tecnológicas e innovadoras, que garanticen la máxima seguridad de los usuarios y la máxima eficiencia de los recursos con unas inversiones lo más ajustadas a las necesidades reales. Utiliza un enfoque integral y multidisciplinar que abarca los campos de la Ingeniería, la Calidad y la Prevención de Riesgos Laborales, siempre con un absoluto y permanente respeto y protección del medio ambiente (Tesicnor, 2016b).

Figura 2. Logotipo de Tesicnor

La aplicación web NOE es un 'Sistema Informático de Ayuda en la Gestión de Emergencias por Inundaciones' que pretende desarrollar el 'Plan de Emergencias para Inundaciones' de una localidad en una aplicación web fiable, útil y sencilla (Tesicnor, 2016a). Es decir, el objetivo de NOE es convertir el documento del 'Plan de Emergencias' en una aplicación web que será una herramienta donde sea más fácil consultar y al mismo tiempo gestionar las acciones. Pero además pretende ofrecer la posibilidad de ir adaptando el plan inicial, que al tratarse de un trabajo teórico realizado en una fecha concreta, no contempla los cambios que puedan darse en el terreno. El objetivo que tiene la aplicación es renovar el plan cada vez que suceda una inundación.

NOE cuenta con tres subsistemas independientes en continua comunicación:

- En primer lugar el sistema de mediciones SAIH de la Confederación Hidrográfica que proporciona automáticamente los caudales y altura de los cauces de los ríos definidos.
- La aplicación de control recoge los datos generados por SAIH, registrándolos y verificando si se cumplen las condiciones para un cambio de Nivel de Emergencia.
- Por último está la interacción con el sistema que puede hacerse bien a través de un navegador Web o a través de una aplicación de escritorio que comprueba periódicamente el estado del Nivel de Emergencia.

La parte con la que interactúa el responsable de gestionar la inundación será la propia aplicación web NOE (figura 3), que dependiendo del nivel de emergencia que indiquen las mediciones SAIH, proporciona la información correspondiente. Esta aplicación se divide en tres apartados principales: Centro de Control, Histórico y Configuración.

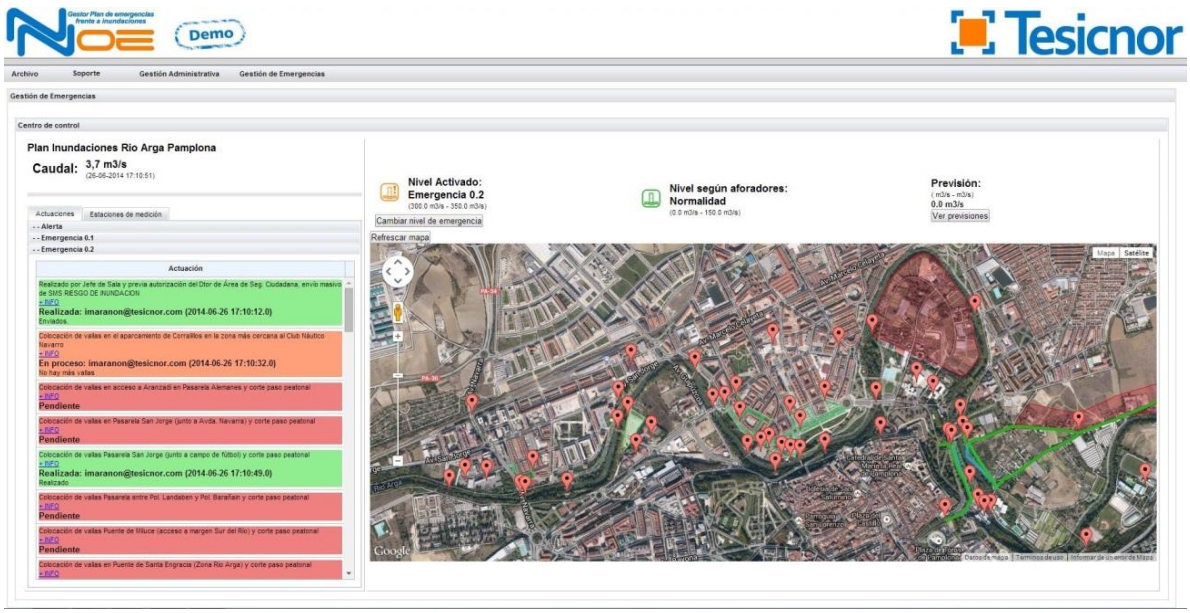


Figura 3. Interfaz de la demo de la aplicación web NOE desarrollada por Tesicnor para Pamplona/Iruña

En el apartado 'Histórico' se visualizan los niveles de emergencia que han sido activados y el estado de las actuaciones lanzadas (realizadas, no realizadas y empezadas pero no cerradas). Por cada elemento activado se registran tanto el usuario que ha realizado la activación como la fecha y los comentarios sobre la acción.

En el apartado 'Configuración' se modela el Plan de Emergencia existente en la aplicación NOE: actuaciones, grupos de emergencia, responsabilidades…

El 'Centro de control' es el punto desde que se visualiza en tiempo real el estado del plan de emergencias. Cuenta con un visor web GIS (del inglés *Geographic Information System*, Sistemas de Información Geográfica) de los puntos de interés para cada nivel así como de las actuaciones a realizar (con todo el grado de detalle al que se quiera llegar: teléfonos, nombres…).

1.3. TECNOLOGÍAS GIS WEB

Las tecnologías GIS web (como es el caso del visor GIS con la que cuenta NOE) resultan de gran ayuda en este tipo de aplicaciones web, ya que proporcionan una visión general del estado de la emergencia (Pusineri, 2004). Se entiende como tecnologías Web GIS a todos aquellos elementos que permiten la representación de cartografía como un contenido más de una página Web. Esto es lo que se engloba bajo la denominación genérica de *Web Mapping* (Olaya, 2013).

El objetivo básico que pretenden cumplir estas tecnologías, especialmente las del lado del cliente, es llevar las funcionalidades de un GIS a la Web, para así compartir la potencia de ambos componentes. Las ventajas

de llevar el GIS a la Web en lugar de incorporar los elementos de esta última en un GIS de escritorio tradicional son notables, y existen grandes diferencias entre las soluciones que se obtienen en ambos casos. Estas diferencias tienen que ver sobre todo con los usuarios y su perfil, así como con el diseño mismo de las aplicaciones.

Algunas de las ideas fundamentales que caracterizan a las tecnologías de *Web Mapping* y su papel actual son las siguientes:

- No es necesario un software GIS específico
- Perfil menos técnico
- Potenciamiento del trabajo colaborativo
- Información más actualizada
- Independencia del sistema
- Personalización de aplicaciones
- Combinación de cartografía y otros elementos

Para que la cartografía web sea lo más parecido posible a un GIS de escritorio, lo más adecuado sería disponer de mapas dinámicos y no de mapas estáticos. Un mapa estático es simplemente una imagen con información cartográfica, la cual no permite ningún tipo adicional de trabajo con ella que no sea la mera observación, no aporta utilidad en el manejo y uso de información geográfica.

Por su parte, un mapa dinámico es aquel que no se compone de una imagen inmóvil, sino que esta varía y se adapta en función de los requerimientos del usuario o según alguna serie de parámetros prefijados. Dentro de los mapas dinámicos se pueden diferenciar entre mapas que son interactivos y los que no, dependiendo de si es el usuario quien directamente modifica la representación del mapa.

En los que no son interactivos (por ejemplo mapas animados que encuadran una determinada zona y muestran la variación de una variable a lo largo del tiempo) tampoco aparecen las funciones esperadas en una aplicación GIS, y una vez más no se requieren tecnologías específicas para poder incorporar este tipo de elementos en una página Web. De este modo, en la mayoría de los casos serán los mapas dinámicos interactivos los más adecuados para la cartografía en la web (Olaya, 2013).

1.4. OBJETIVO DEL TFM

Por lo tanto, el objetivo de este Trabajo Fin de Máster (TFM) es mejorar el visor GIS de la aplicación de tal manera que el resultado final sea un mapa dinámico e interactivo lo más intuitivo posible donde se podrá visualizar la información de todas aquellas capas necesarias a la hora de gestionar una situación de emergencia a causa de una inundación. Para ello se diseñará y desarrollará un sistema de carga de datos geolocalizados afectados por las inundaciones (áreas inundables, inmuebles afectados, calles cortadas…) y su inclusión en herramientas GIS libres que puedan ser integrables en la aplicación web NOE. Los datos calculados podrán servir tanto para representarlos en la aplicación como información auxiliar para redactar un nuevo 'Plan de Actuaciones'.

La aplicación actual está montada sobre un servidor Tomcat, y el mapa se ha desarrollado con la ayuda de la API de Google Maps. Los marcadores y polígonos que se muestran en el visor y que hacen referencia a las acciones que se deben realizar están guardadas en una base de datos MySQL sin utilizar ninguna extensión espacial (MySQL Spatial), es decir, funciona con BLOBs (Binary Large Objects), con archivos en formato KML. Estos KML fueron creados a mano sobre el Google Earth utilizando imágenes aéreas de la inundación sucedida en Pamplona en junio de 2013.

Además, este TFM pretende servir como guía en la parte de la carga de datos geolocalizados para que pueda ser útil en un futuro para la empresa Tesicnor, y en especial para los desarrolladores de la aplicación web NOE. Mediante anexos se explicarán todos los pasos que se deben dar para la carga de datos geolocalizados independientemente del municipio o elementos que deban aparecer.

2. MATERIALES Y MÉTODOS

2.1. MATERIALES

En el desarrollo de cualquier aplicación *Web Mapping* es indispensable contar por un lado con las herramientas GIS, dentro de una arquitectura web GIS (Evangelidis et al., 2014), y por otro lado con los datos espaciales que aparecerán en el mapa que se vaya a diseñar.

Para poder saber qué herramienta se integra mejor en la aplicación web NOE se ha elaborado un estado del arte de los productos existentes y dependiendo de las funcionalidades que aportan junto a las necesidades de la propia aplicación y sus desarrolladores se ha escogido el más adecuado. El objetivo es montar toda la arquitectura web GIS necesaria para poder llevar a cabo la metodología que se expondrá en el apartado de la metodología.

En cuanto a los datos, en esta primera parte se pretende presentar distintas fuentes de datos que estén relacionados de alguna u otra forma con las inundaciones y los elementos afectados como pueden ser calles, edificios… El objetivo es conocer qué elementos de la realidad se encuentran afectados por una mancha de inundación.

2.1.1. Herramientas GIS

Para entender cómo funcionan estas herramientas y qué aportan, hace falta saber que el proceso de desarrollo de un mapa web se divide en tres etapas (Smith, 2016): en la primera se recopilan datos y se realizan las tareas de geoprocesamiento preparatorios (escritorio), en la segunda etapa los datos se cargan y se utilizan para crear un servicio de mapas (escritorio a servidor) y la tercera etapa es esencialmente un proceso de diseño web donde se desarrolla la interfaz utilizando una librería de cliente *Web Mapping* para manejar la integración entre el navegador (*browser*) y el servicio de mapeo. Esta última etapa es la que se va a estudiar en los siguientes párrafos, ya que son las APIs (Application Programming Interfaces) donde se apoyan la mayoría de los desarrolladores de aplicaciones web que manipulan la información georreferenciada (Fernandes, Goulão, & Rodrigues, 2013), y dependiendo de esta librería se definirá la arquitectura web GIS necesaria.

Para poder buscar y comparar información sobres distintas herramientas GIS libres, lo primero es hacer una lista lo más completa posible de todos los productos para luego poder analizar una a una. Pero en este caso la gran cantidad de herramientas existente (Ballatore et al., 2011) dificulta la tarea de comparación y por lo tanto se ha optado por elegir las herramientas más usadas o más documentadas y hacer un estudio solamente sobre esas.

Las herramientas que en un principio se han considerado han sido: Google Maps API, OpenLayers, Cesium, MapBox, Leaflet, D3, Potree, Bing Maps, Apple Maps, Here Maps, CartoDB, la API de SITNA, MapQuest, MapBender, CloudMade, ArcGIS OnLine de ESRI, QGIS Cloud, MangoMap, GeoMoose, GeoXt, Cartaro… La idea es escoger de entre todas estas las 5 más usadas o populares para analizarlas.

Para empezar a conocer el uso de cada una de ellas, el primer paso ha sido utilizar la herramienta de tendencias de Google[1]. Los gráficos resultantes (figura 4 y figura 5) indican la tendencia de búsquedas para los términos clave que se le indique. Debe tenerse en cuenta que esta herramienta trabaja con valores relativos, comparando un término de búsqueda con otro sin proporcionar datos de valores absolutos de la cantidad exacta de búsquedas.

Por ese motivo, se ha partido de las 6 APIs que a priori parecen ser los más conocidos según se ha podido observar en los blogs y páginas web consultadas: Google Maps API, OpenLayers, MapBox, Leaflet, CartoDB, y ArcGIS OnLine de ESRI. De entre estas seis herramientas se escogerá basándose en la figura 4 el menos popular para comparar el resto de las herramientas con este. Si alguna herramienta lo supera significará que se trata de otra herramienta a tener en cuenta.

[1] Fecha de consulta: mayo de 2016.

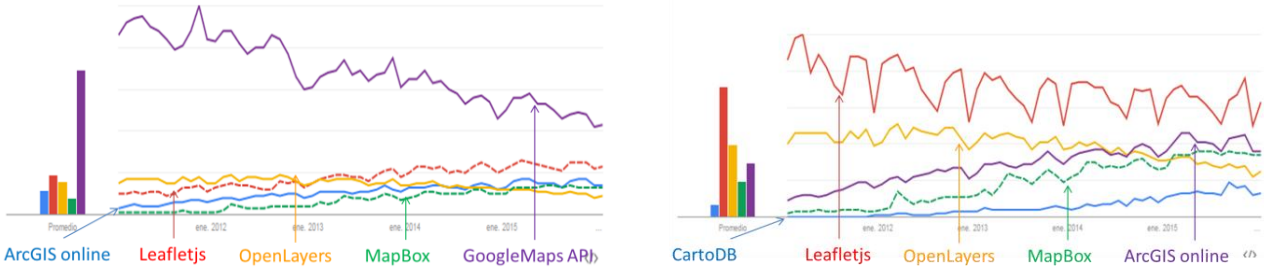


Figura 4. Tendencias de búsqueda de las herramientas *Web Mapping* más conocidas a priori

Claramente, aunque su influencia disminuya, la API de Google Maps es la que mayor éxito ha tenido en los últimos 5 años y CartoDB el que menos, aunque sea una herramienta que cada vez se utilice más. Por lo tanto el resto de las herramientas se compararán con esta última y así saber si merece la pena analizarla:

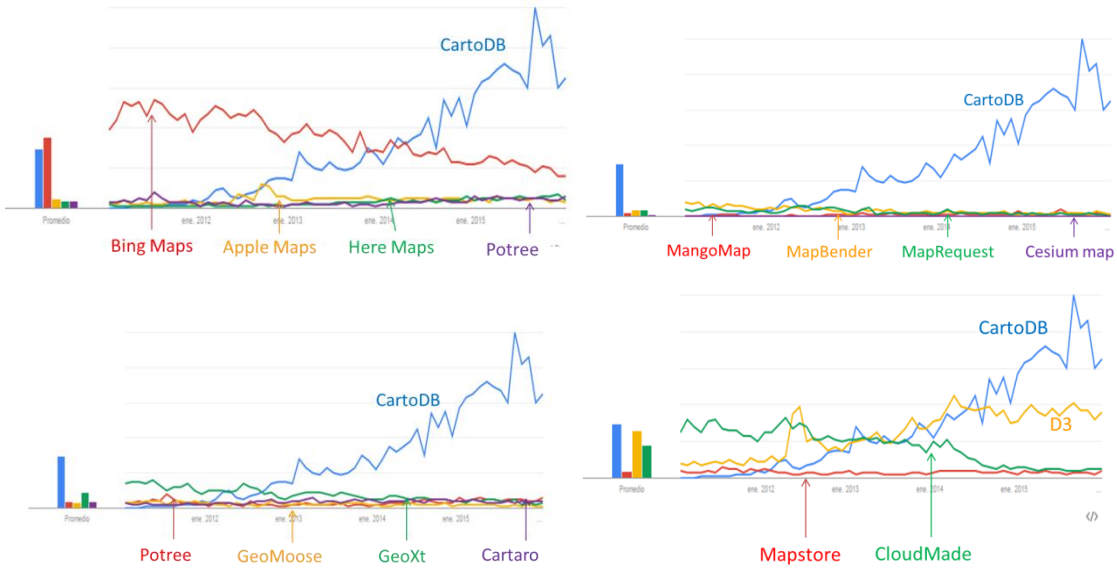


Figura 5. Tendencias de búsqueda de distintas herramientas *Web Mapping* para determinar si alguno se usa más que CartoDB

Con los gráficos de la figura 5 se puede concluir que entre las herramientas restantes, solamente Bing Maps API y D3 (que al final se trata de una librería de JavaScript para producir, a partir de datos, infogramas dinámicos e interactivos donde cabe la posibilidad de trabajar con datos espaciales) se acercan a los valores del CartoDB pero sin superarlo, y por lo tanto son las herramientas que desde un principio se han considerado como las más usadas las que mejores resultados obtienen. Que la tendencia sea ascendente es otra de las características a tener en cuenta, ya que esto indica que en un futuro, por lo menos cercano, su mantenimiento está garantizado.

Aun así, no es suficiente con que sean las herramientas más buscadas, y es necesario saber si la documentación o SDK (del inglés, *Software Development Kit*, Kit de Desarrollo de Software) disponibles son suficientes o los foros de desarrolladores son lo suficientemente activos como para poder consultar y mejorar la aplicación a medida que se vaya trabajando. El objetivo es que la herramienta seleccionada tenga el apoyo de los desarrolladores y que se asegure su continuidad en el tiempo.

Para poder conocer el estado de cada una de las herramientas, en los siguientes puntos se resumen las características principales: última versión estable (y actualizaciones), documentación existente, SDK, foros, herramientas/APIs y sus funcionalidades.

El servicio de ArcGIS online de ESRI está integrado con el software de escritorio ArcGIS para facilitar la carga de datos e incluye muchas funciones de análisis espacial que en otros servicios en la nube no existen.

Sin embargo, hay algunos problemas que dificultan su uso, ya que es menos flexible que los servicios de código abierto y el esquema de precios está destinado a grandes empresas, son softwares privativos (Smith, 2016). Por este motivo se ha descartado su utilización desde un principio, ya que difiere del objetivo establecido desde la propia empresa de que la herramienta a utilizar sea libre.

2.1.1.1. Google Maps API

La API para JavaScript de Google Maps ofrece a desarrolladores varias formas de insertar mapas de Google en páginas web (Google, 2016). La API es un servicio gratuito, pero sujeta a unos términos y condiciones (tabla1). La mayoría de los sitios web y las aplicaciones pueden usar la API para JavaScript estándar sin cargo. No obstante, si se genera constantemente una gran cantidad de tráfico, se aplicarán los límites de uso y se deberá pagar por el uso adicional. Además, al no ser de código Open Source, cualquier tipo de cambio en las condiciones de uso puede acarrear problemas en el negocio. Aun así sigue siendo uno de los más utilizados.

Tabla 1. Condiciones de uso de la API para Javascript de Google Maps

Límites de uso estándar (usuarios de la API estándar)	
Gratis hasta exceder 25.000 cargas de mapas por día durante 90 días consecutivos	Habilitar la facturación de pago según el uso para desbloquear cuotas mayores: • Al exceder los límites de uso gratuito, se facturarán USD 0,50 por 1.000 solicitudes adicionales, hasta 1.000.000 diarias

Límites de uso Premium (clientes de Google Maps API for Work)	
Precio basado en el volumen requerido	Beneficios adicionales de un plan Premium: • Contratos anuales con términos corporativos • Soporte técnico las 24 horas • Acuerdo de nivel de servicio (SLA) • Licencias para casos de uso interno, de OEM y de seguimiento de recursos

Además de la API JavaScript para la web, también se dispone de APIs de Google Maps para servicios web: 'Google Maps Geocoding API', 'Google Places API Web Service', Google Maps Elevation API', Google Maps Distance Matrix API', 'Google Maps Roads API', 'Google Maps Time Zone API', Google Maps Geolocation API' y 'Google Maps Direction API'.

Las funciones que se consiguen con todas ellas son entre otras:

- Imágenes de 45 grados
- Autocompletado de direcciones
- Analytics: obtener un conocimiento del modo en que los visitantes del sitio web interactúan con los mapas
- Búsqueda de negocios y de lugares de interés
- Indicaciones
- Matrices de distancia
- Predicción de la duración del trayecto
- Ruta del tráfico
- Herramientas de dibujo
- Perfiles de elevación
- Asistencia para empresas
- Codificación geográfica normal e inversa
- Imágenes globales por satélite

- Procesamiento KML
- Actualizaciones rápidas desde Map Maker
- Tráfico en tiempo real
- Fiabilidad total
- Static Maps
- Street View
- Mapas con estilos
- Compatibilidad con diferentes dispositivos

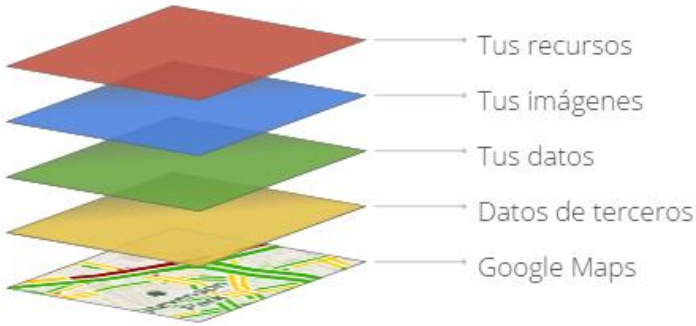


Figura 6. Diagrama de capas que se pueden insertar en la API de Google Maps

Los únicos formatos con los que trabaja son el KML/KMZ y el GeoRSS.

La última versión estable es el 3.23 (noviembre de 2015) aunque ya ha salido el 3.24 de prueba en abril de 2016. Dispone de una documentación completa, un SDK y un foro en el 'Stack Overflow' y otro en el 'Google Groups'.

2.1.1.2. Openlayers

OpenLayers es una librería escrita en JavaScript, orientado a objetos, que facilita acceder, manipular y mostrar mapas en cualquier página web o aplicación, del lado cliente, por lo tanto sin necesidad de un servidor ni configuración (Baena Carrillo & Ferré, 2011). Proporciona una API que permite la creación de clientes web para acceder y manipular información geográfica proveniente de muy variadas fuentes (OpenLayers, 2016):

- Web Map Service
- Web Feature Service
- Mapas comerciales: Google Maps, Bing, Yahoo
- Información genérica vectorial: EsriJSON, GML, XML, GPX, GeoJSON, KML, MVT, TopoJSON, Shapefile…
- Mapas de OpenStreetMap
- ...

Es un proyecto del Open Source Geospatial Foundation (OSGeo) y la librería está en puro JavaScript, de uso totalmente libre bajo licencia BSD. Es la librería que más cerca de las funciones trabaja, es decir, es una de las librerías fundamentales que permite controlar y trabajar con la mayor parte de funciones espaciales existentes, lo que hace que las funcionalidades sean mayores. Otras librerías que son más sencillos se apoyan en librerías como ésta para sus funcionalidades (capas de abstracción), disminuyendo las posibilidades y dificultando el desarrollo y diseño de la aplicación web. Dependiendo del uso que se le quiera dar a la herramienta interesará escoger una que sea más fácil u otra que sea más completa. El esquema de trabajo para este tipo de librerías es el siguiente:

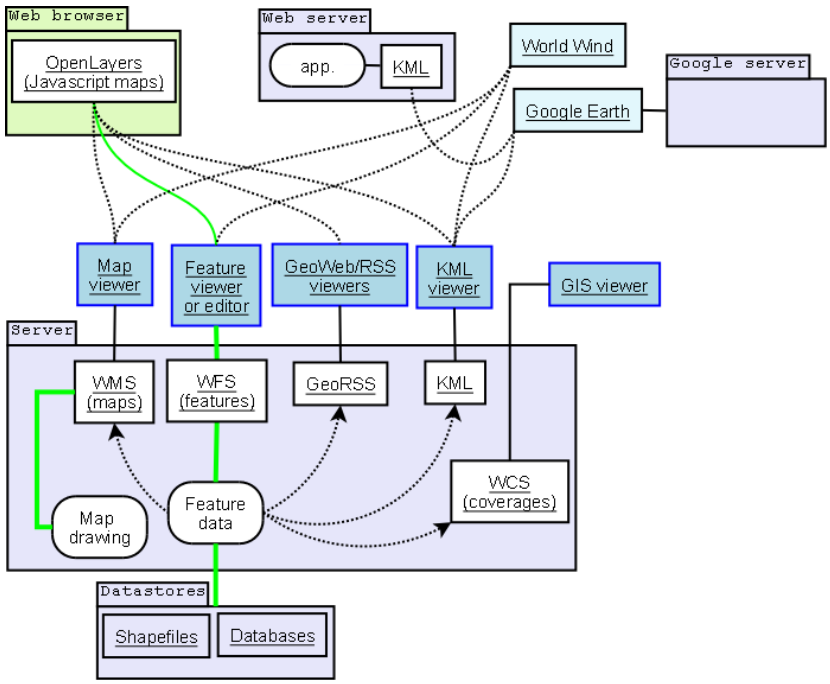


Figura 7. Esquema de trabajo con la librería de OpenLayers

Recientemente se acaba de liberar la versión de OpenLayers 3.2.0. Esta versión 3 aún se encuentra bajo un fuerte desarrollo, y como es una nueva arquitectura respecto a la versión 2, aún faltan muchas características que sí existían en la versión 2.

Sin embargo OpenLayers carece de una rica interfaz de usuario. Para solucionar esto es posible contar con la librería Ext JS. GeoExt (GeoExt, 2016) combina los controles geoespaciales de OpenLayers con los componentes de interfaz de usuario de Ext JS en un framework que permite construir aplicaciones GIS de estilo similar a las de escritorio, pero en un navegador.

Openlayers dispone de una documentación completa en su página web oficial o que puede ser descargada junto a la API (todo en formato zip), existe un SDK que se instala con el OpenGeo Suite y un foro activo en el 'Stack Overflow'.

2.1.1.3. Leaflet

Leaflet es una librería Javascript Open Source muy ligera (33 Kbs) para trabajar con mapas interactivos mobile-friendly en OpenStreetMap. Es un proyecto del Open Source Geospatial Foundation (OSGeo) y no tiene limitaciones en su uso (Leaftlet, 2016).

Su API se caracteriza por la sencillez y el rápido aprendizaje. Al ser una librería muy ligera cubre solamente las características básicas aunque estas funcionen a la perfección. Lo que tiene es que es ampliable mediante plugins y así conseguir mayor número de funcionalidades. En ella se basan otras herramientas como pueden ser el CartoDB o el Mapbox.

Preparado para soporte móvil, otra de las ventajas es que funciona tanto en los modernos como en los viejos navegadores web. Trabaja con HTML5 y CSS3.

Soporta:

- GeoJSON
- KML, CSV, WKT, TopoJSON, GPX: con el plugin Leaflet-Omnivore (entre otros). No soporta GML.
- Web Map Service. No soporta el Web Feature Service aunque existen plugins '3rd party'

Es una API bien documentada cuya última versión estable es el 0.7.7, actualizada en octubre de 2015, aunque está en progreso el 1.0 (liberado en abril de 2016). No dispone de SDK y el foro comunitario oficial para los desarrolladores se encuentra en 'Google Groups'.

2.1.1.4. CartoDB

CartoDB es una librería *open source* construida sobre Leaflet que trabaja en la nube, cuya potencia radica en el almacenamiento de las tablas en una base de datos PostgreSQL, con el componente espacial PostGIS. Esto permite realizar geoprocesos de manera muy rápida, gracias a la utilización de índices espaciales (CartoDB, 2016). Se podrían diferenciar dos grandes grupos de herramientas: la plataforma de CartoDB y el Editor.

La plataforma CartoDB permite llevar a cabo operaciones GIS, análisis de datos y visualización de datos, proporcionando un conjunto de herramientas y APIs. De hecho, el Editor de CartoDB (CartoDB Editor) es una aplicación construida en la parte superior de la Plataforma CartoDB. Las APIs y herramientas disponibles son:

- CartoDB.js: esta librería Javascript es la herramienta principal para crear visualizaciones.
- SQL API: permite interactuar (insertar, actualizar, borrar, seleccionar…) con las tablas o datos dentro del CartoDB como si se estuviera ejecutando instrucciones SQL contra una base de datos normal.
- Maps API: permite generar mapas basados en datos alojados en su cuenta CartoDB y aplicar SQL personalizado y CartoCSS a los datos. El API genera una URL basada en XYZ para recuperar capas proyectadas en Web Mercator usando clientes web tales como Leaflet, Google Maps u OpenLayers.
- Import API: permite subir archivos a una cuenta CartoDB, comprobar su estado de carga actual, así como eliminar y enumerar los procesos de importación en una cuenta determinada. Esta API se compone de varias peticiones HTTP dirigidas a un conjunto de criterios de valoración CartoDB que tienen que ver con la conversión y la importación de los archivos enviados. Las tablas CartoDB se pueden clasificar como Tablas Estándar o Tablas Sincronizadas. Además, el servidor ArcGIS Conector le permite importar capas de ArcGIS en sus bases de datos.
- Data Services API (sujeto a limitaciones de cuota): ofrece un conjunto de servicios basados en la localización, que se pueden utilizar para personalizar subconjuntos de datos en la visualización programáticamente.
- CartoCSS: es el lenguaje de sintaxis que permite personalizar el estilo de los datos del mapa. Similar a trabajar con un CSS para el diseño de páginas web, CartoCSS es específico para diseñar aquellas propiedades relacionadas con los controles que puedan aparecer en un mapa (tamaño del marcador, color del marcador, trazo de la línea, la visualización de texto, etc.). Se pueden aplicar estilos CartoCSS directamente desde el Map View del Editor de CartoDB (CartoDB Editor). Opcionalmente, también es posible aplicar CartoCSS con CartoDB.js y el Maps API.
- Torque.js: es un método de representación eficiente y elegante para animar los datos. Torque.js utiliza TileCubes, que son representaciones JSON de datos multidimensionales con las coordenadas geoespaciales, para renderizar datos en el cliente.

El Editor de CartoDB es una herramienta de autoservicio (self-service) en línea para mapeo y análisis que combina una interfaz intuitiva con otras funciones de gran alcance. Posibilita mezclar y combinar los conjuntos de datos para extraer nuevos conocimientos en las visualizaciones. El interfaz gráfico permite tanto diseñar y analizar como publicar APIs. Soporta formatos de archivo CSV, SQL, XLS, GeoTIFF, GTFS, GeoJSON, KML, ESRI Shapefile y GPX.

Existe una extensa documentación sobre la librería y el uso de las herramientas, además, también dispone de SDK. La última versión estable es el 3.15.9 de febrero del año 2016 y se encuentran foros de desarrolladores tanto en 'Stack Overflow' como 'Google Groups'.

2.1.1.5. MapBox

Mapbox es una plataforma de desarrollo para la creación y uso de mapas que trabaja en la nube. Está dividido en bloques de herramientas que soportan todos los aspectos relacionados con el proceso de elaboración de mapas web y móviles (MapBox, 2016). Permiten:

- Diseño y estilo del mapa: MapBox Studio (TileMill lleva 2 años sin actualizar).
- Cargar datos personalizados o utilizar conjuntos de capas del propio MapBox:
    - Mapbox Streets, Mapbox Terrain y Mapbox Satellite, alimentados por OpenStreetMap
    - Formatos soportados son mayoritariamente vectoriales: CSV, ESRI Shapefile, GeoJSON, KML, GPX, GeoTIFF, SQLite, PostGIS.
- Crear y publicar aplicaciones web con todas las funciones para la interacción con el cliente: Mapbox GL JS, Mapbox JS y Mapbox Editor.
- Crear mapas estáticos mediante programación: MapBox Static API.
- Desarrollar aplicaciones móviles nativas para iOS y Android: se proporcionan SDK para ambos.
- Ampliar la funcionalidad de las aplicaciones con servicios web de geocodoficación (Geocoding API), direcciones (Directions API), distancias (Distance API), análisis espacial (Turf.js)…

Mapbox.js es un plugin de Leaflet. Extiende las funcionalidades de Leaflet con código adicional para integrar los servicios y los datos en Mapbox. El archivo Mapbox.js por defecto incluye una copia de una versión estable particular de Leaflet.

En cuanto al editor, TileMill permitía exportar las capas con un estilo propio como .mbtiles y alojarlas en otro sitio. Pero el programa no ha sido actualizado en 2 años y solamente se encuentra en activo el MapBox Studio, con el que los estilos personalizados se deben alojar en el propio MapBox (nube).

Al igual que las herramientas anteriores, la documentación es amplia y completa con distintos SDK disponibles. La última versión es la 3.12.4 de marzo de 2016 y un foro de desarrolladores muy activo en 'Stack Overflow'.

2.1.1.6. Comparación y elección de la herramienta

Para la comparación de distintas APIs, la mayoría de los estudios se basan en preguntas realizadas a conjuntos de usuarios de estas APIs (Fernandes et al., 2013), pero en este caso se ha intentado compararlos en base a sus capacidades y herramientas disponibles junto a una pequeña muestra de cómo se utilizan cada una de ellas (Anexo I).

Para la comparación se ha creado y rellenado la tabla 2 donde se pueda diferenciar las características de cada una de las herramientas analizadas. Estas características son las definidas por los desarrolladores de Tesicnor, que en un principio son las propiedades que más interesan: documentación, última versión, existencia de SDK, herramientas/APIs disponibles, formatos geográficos que soportan, si son o no de código abierto, cuáles trabajan en la nube y las licencias.

Se ha visto que la arquitectura web GIS que necesita cada una de las herramientas es distinta, pudiendo diferenciar dos grandes grupos: los servicios 'self-hosted' de *Web Mapping* y los servicios en la nube (Smith, 2016). La primera opción (figura 8) implica alojar los datos en un servidor propio o de la empresa/institución. Es un enfoque potente y flexible debido a la gran variedad de software de código abierto (FOSS por sus siglas en inglés Free Open Source Software) de alta calidad disponible, aunque es exigente en términos de habilidades técnicas. La segunda opción implica la posibilidad de subir los datos de mapas a un servicio en la nube, lo que evita la necesidad de tener un servidor propio y un gestor de bases de datos, pero normalmente se introducen tasas mensuales (Smith, 2016).

Tabla 2. Tabla comparativa de las herramientas analizadas

	DOCUMENTACIÓN	HERRAMIENTAS DISPONIBLES	ÚLTIMA VERSIÓN ESTABLE	SDK	EDITOR ONLINE	FORMATOS	SERVICIOS WEB OGC	OPEN SOURCE	LICENCIA/PRECIOS
GOOGLE MAPS API	Completa	API de JavaScript y APIs de servicio web: Geocoding, Places, Elevation, Distance, Roads, Time Zone, Geolocation y Direction	Noviembre 2015	iOS	No	KML/KMZ y GeoRSS	WMS, WMTS *1	Al no ser de Open Source, pueden aparecer problemas si cambian las condiciones de uso	Gratuito, pero sujeta a unos términos y condiciones
OPENLAYERS	Completa	API: Un único zip (con la documentación un total de 3540 archivos)	Abril 2016*2	Se instala con el OpenGeo Suite	No	EsriJSON, GML, XML, GPX, GeoJSON, KML, MVT, TopoJSON, ESRI Shapefile	WMS, WFS, WFS-T, WCS, WMTS, TMS, WPS, SOS, CSW	Open Source	Libre
LEAFLET	Completa	Una única librería de 33 Kb, ampliable con plugins	Octubre 2015	No	No	GeoJSON, KML, CSV, WKT, TopoJSON, GPX	WMS, WFS, WFS-T, WMTS, TMS	Open Source	Libre
CARTODB *3	Completa	CartoDB.js, SQL API, Maps API, Import API, Data Services API, CartoCSS, Torque.js y Editor	Febrero 2016	Android, iOS y Windows	Sí	CSV, SQL, XLS, GeoTIFF, GTFS, GeoJSON, KML, ESRI Shapefile y GPX	WMS, WMTS, WFS *1	Open Source	Gratuito: 4 tablas en 250MB 149$/mes: 5 tablas en 500MB 499$/mes: 6 tablas en 1.5GB 825$/mes; +6 tablas en 5 GB Visitas ilimitadas
MAPBOX *3	Completa	MapBox Studio, MapBox Editor, Mapbox GL JS, Mapbox JS, Mapbox Static API, Geocoding API, Directions API, Distance API, Turf.js…	Marzo 2016	iOS y Android	Sí *4	CSV, ESRI Shapefile, GeoJSON, KML, GPX, GeoTIFF, SQLite, PostGIS	WMS *1	Open Source	Gratuito: 10 estilos en 1GB 49$/mes: 20 estilos en 5GB 499$/mes: 30 estilos en 50GB Enterprise:+50 estilos Límites de visitas según licencia (50.000, 100.000, 1.000.000 por mes)

*1 No ha sido posible encontrar más información, lo que no significa que estos sean los únicos.

*2 La versión 3 se encuentra bajo un fuerte desarrollo, faltan muchas características que sí existían en la versión 2.

*3 En la nube.

*4 También editor de escritorio: MapBox Studio Classic

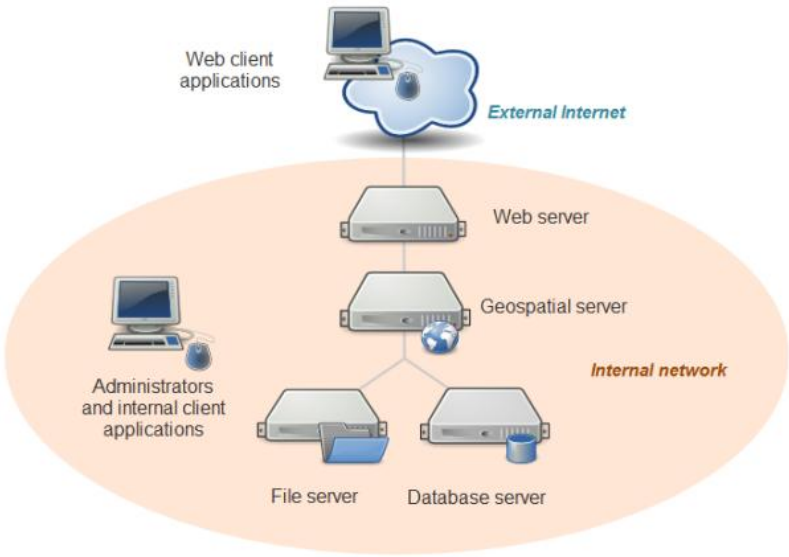


Figura 8. Esquema de la arquitectura web GIS para servicios 'self-hosted'

Dentro del enfoque del 'self-hosted' hay una importante distinción entre datos de mapas pre-renderizadas y 'on-the-fly'. Los datos de mapa normalmente se entregan usando 'tiles' (una cuadrícula regular de imágenes georreferenciadas, baldosas) pre-renderizadas mediante formatos de imágenes comprimidas para la entrega rápida al navegador del cliente. Para las plataformas de mapeo con muchas capas de datos esto se traduce en grandes demandas de almacenamiento en el servidor. En renderizaciones on-the-fly esto se evita mediante la creación de servidores de mapas ('Map Server') que renderizan el mapa en tiempo real (Smith, 2016).

Dentro de este enfoque la mayoría de aplicaciones líder que se han desarrollado son herramientas de código abierto, incluyendo PostGIS/PostgreSQL para el almacenamiento de datos espaciales, Mapnik para la renderización de mapas y dos de las librerías del lado del cliente para *Web Mapping* más populares actualmente, OpenLayers y Leaflet, todos ellos muy ligados a OpenStreetMap (OSM). Mientras tanto, la API de Google Maps fue el origen del *Web Mapping* en la nube, que a pesar de que en un principio las funcionalidades eran limitadas ha mejorado recientemente sus opciones de personalización con servicios como My Maps, Fusion Tables y la API para JavaScript. En su lugar, los recientes avances en los servicios de mapeo en la nube han sido dirigidos por nuevas empresas, principalmente por MapBox y CartoDB (Smith, 2016).

En base a lo realizado en el Anexo I, de entre estos dos enfoques OpenLayers y CartoDB son las herramientas que mejor sensación transmiten en cada una de ellas a la hora de trabajar:

- Las funcionalidades de la API de Google Maps son limitadas y está muy ligado al formato KML, toda capa se tendría que convertir a este formato.
- Mapbox es una herramienta muy potente para diseño y estilo de mapas pero muy poco intuitivo para este tipo de trabajos, donde se pretende mostrar sobre un mapa base las capas que se vayan a crear. Además, por lo que se ha podido comprobar, las herramientas disponibles tienden a saturarse con archivos pesados.
- De entre los servicios en la nube, CartoDB es la herramienta que más facilidades aporta, con un modo de trabajo claro e intuitivo. Fácil de integrar en una aplicación web con muy poco código ejecutable, la mayor parte del proceso se realiza en el editor en línea, cuyas funcionalidades se asemejan a la de un GIS de escritorio, salvando las distancias.
- Entre OpenLayers y Leaflet ambas aportan las funcionalidades necesarias, aunque con Leaflet sea necesario conocer los plugins que se necesitan en cada operación. Aun así, conociendo que OpenLayers es la librería que más cerca de las funciones trabaja, se ha decantado por esta opción.

Por lo tanto la última elección se realizará en base a la siguiente pregunta: ¿se prefiere alojar todo en la nube pagando y sin acceso a las herramientas que la componen, o alojar todo en el servidor de la empresa y mantener toda la arquitectura aunque esto implique más habilidades técnicas?

Al final se ha optado por montar toda la arquitectura en un servidor propio, ya que la empresa dispone de recursos suficientes como para poder manejar y gestionar una arquitectura de este tipo y por tanto no supone un esfuerzo en cuanto a habilidades técnicas a la plantilla. Esto quiere decir que la librería que se utilizará es el OpenLayers, con todas las ventajas que ello supone (tabla 2): es gratuito y no depende de cuántas capas se carguen o si en un futuro se cambien las condiciones de uso, cuenta con el soporte de toda una comunidad de desarrolladores que mantienen el código como una de las librerías más utilizadas y por último se podrá disponer de mayor flexibilidad y con mayor número de posibles soluciones ya que es la librería que más funcionalidades aporta.

2.1.1.7. Arquitectura web GIS

OpenLayers, al trabajar en el lado del cliente (figura 9), necesita tener acceso a un servidor de mapas. Es decir, al ser un Web Map Client una de las principales tareas que realiza es conseguir imágenes de mapas de un servidor. El cliente pide al servidor de mapas aquello que quiere ver y une todas las imágenes que recibe de forma que obtiene el mapa tal y como el usuario lo visualiza. Cada vez que navega o hace zoom en el mapa, el cliente tiene que hacer nuevas peticiones. Este servidor de mapas será un Web Map Server, que cumple con los estándar OGC[2] (Baena Carrillo & Ferré, 2011). Si los Web Map Server a utilizar son los mapas de OpenStreetMap, Google, Bing o del propio IDENA sería suficiente con conocer su URL, pero en este caso lo que se pretende es crear nuestras propias capas y servírsela a OpenLayers. Por este motivo será necesario construir una arquitectura adecuada.

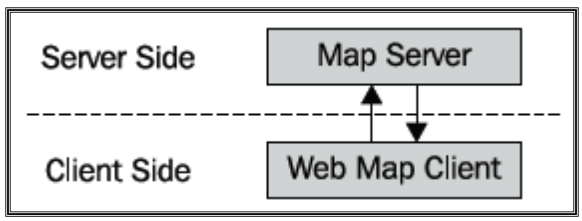


Figura 9. Modelo cliente/servidor

Existe una gran variedad de tipos de softwares GIS que se adaptan a las tareas solicitadas. Para simplificarlo se pueden agrupar en cuatro categorías: bases de datos espaciales para el almacenamiento, la publicación de datos espaciales para el intercambio (Web Map Server), librerías 'mapping' para la visualización de conjuntos de datos espaciales (Web Map Client), y el análisis espacial para geoprocesamientos y algoritmos espaciales (Swain et al., 2015).

En este caso la librería ya ha sido seleccionada y el análisis espacial se realizará antes de introducir los datos espaciales en la base de datos correspondiente mediante un GIS de escritorio, en este caso QGIS que es uno de los FOSS de GIS más utilizados, con un interfaz claro y una de las mayores comunidades de usuario, además compatible con otros GIS como puede ser GRASS GIS (Steiniger & Hay, 2009). Por lo tanto se analizarán las bases de datos y softwares de publicación de código abierto más utilizados.

Las bases de datos espaciales almacenan los datos en un sistema de archivos que son adecuados para grandes conjuntos de datos y disponen de mecanismos eficientes para el almacenamiento, consulta, análisis y actualización de estos datos. La mayoría de bases de datos espaciales son extensiones de bases de datos de lenguajes de consultas estructuradas (Structured Query Language, SQL) ya existentes e implementan estándares OGC-SFS, que define cómo se deben representar los objetos espaciales. Los más conocidos son (Swain et al., 2015):

[2] El Open Geospatial Consortium (OGC) trabaja en la definición de especificaciones públicas para el intercambio de datos entre GIS. Definen estándares abiertos e interoperables dentro de los GIS y de la World Wide Web que facilitan el intercambio de la información geográfica entre distintos sistemas (Song et al., 2008).

- PostGIS de PostgreSQL: es el más utilizado de todos debido a sus funcionalidades (geoprocesos, transformaciones entre sistemas de referencia, conversiones entre vectoriales y ráster…) y formatos que soporta.
- SpatiaLite de SQLite: no toma en cuenta los sistemas de referencia de los datos y sus capacidades van dirigidas más a usuarios particulares que a soportar múltiples conexiones simultaneas como sucede en las aplicaciones web.
- MySQL Spatial de MySQL: a pesar de que MySQL es la herramienta de código abierto (FOSS) más popular en cuanto a bases de datos, su extensión para datos espaciales todavía no alcanza las funcionalidades del PostGIS.

Los servidores de datos geoespaciales hacen que los datos espaciales estén disponibles en un formato adecuado para la web (inweb-friendly format), que en el caso de la información geográfica esto lo posibilita los estándares de la OGC para servicios web, principalmente el Web Map Service (OGC-WMS), el Web Feature Service (OGC-WFS) y el Web Coverage Service (OGC-WCS). Los FOSS más utilizados para este fin son (Swain et al., 2015):

- MapServer
- GeoServer
- deegree

En este caso tanto MapServer como GeoServer son los más populares y son detalles los que diferencian el uno del otro. Geoserver proporciona un interfaz web para la configuración de datos en el servidor haciéndolo más fácil de usar que MapServer. Sin embargo, MapServer se puede configurar mediante programación a través de MapScript en distintos entornos de desarrollo incluyendo, PHP, Python, Perl, Ruby, Java y .NET. GeoServer se puede configurar mediante programación a través de APIs REST. En términos de rendimiento, MapServer tiende a superar a GeoServer. Para terminar, este último proporciona una implantación WPS que se puede utilizar para geoprocesamiento (Swain et al., 2015).

Una manera de tener todas las herramientas instaladas en el servidor es utilizando una plataforma geoespacial completa como puede ser el OpenGeo Suite (Daras, Agard, Cambazard, & Penz, 2015) que integra programas de todos los campos de los GIS como se observa en la figura 10. Esta plataforma permite gestionar datos y construir mapas y aplicaciones a través de navegadores web, ordenadores de mesa y dispositivos móviles. Construida sobre softwares geoespacilaes de código abierto (PostGIS+GeoServer+OpenLayers+QGIS), OpenGeo Suite cuenta con una arquitectura robusta y flexible que permite a las organizaciones gestionar y publicar datos geoespaciales de manera fiable (Boundless, 2016).

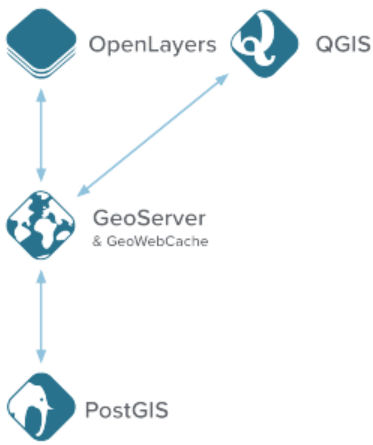


Figura 10. Herramientas que componen OpenGeo Suite y sus relaciones

En resumen, todo el conjunto de herramientas y software disponibles con OpenGeo Suite y los softwares que se utilizan junto a ellos (*) suelen ser (Daras et al., 2015):

- PostGIS es una extensión que convierte el sistema de base de datos PostgreSQL en una base de datos espacial.
- GeoServer permite a los usuarios compartir y editar datos geoespaciales. Posibilita publicar datos espaciales de las fuentes más importantes utilizando estándares abiertos.
- OpenLayers es una librería JavaScript para construir aplicaciones de mapas en un navegador.
- QGIS* es un conocido GIS de escritorio con la que interactuar con PostGIS es más fácil.
- R* es un entorno de software libre para computación y gráficos estadísticos con extensiones para el análisis espacial.
- PgAdmin* es una interfaz de administración de base de datos PostgreSQL que puede hacer frente a la base de datos PostGIS.

Pero al ser este todavía un prototipo y no tratarse de una aplicación que se implante de inmediato, de momento se ha decido instalar las herramientas independientemente en el servidor y en un futuro, si se ve que el rendimiento de las herramientas es óptima y que se quiere seguir trabajando con ellas, volver a plantearse si merece la pena o no montar todo el conjunto de OpenGeo Suite y pagar por el soporte que ofrecen.

La instalación de las herramientas GIS que se han escogido y sus características (sobre todo la versión) se especifican en la tabla 3. Todas las instrucciones que se describan en el informe como en los anexos se han probado con estas herramientas.

Tabla 3. Herramientas instaladas para desarrollar el prototipo

Herramienta	Seleccionada	Características
Servidor	Amazon AWS - EC2	- Hospedaje de servidores virtuales en la nube - Instancia t2.small: 12 créditos por hora de la CPU y 2 GiB de memoria
Software GIS de escritorio	Quantum GIS (QGIS)	- Software instalado en el propio ordenador - Versión QGIS 2.8.9-Wien
Base de datos espacial	PostgreSQL con extensión PostGIS	- Instalados en el servidor - Versiones: PostgreSQL 9.4.5 PostGIS 2.2.0 - Se administran desde entorno de escritorio mediante el software pgAdmin III
Servidor de mapas	GeoServer	- Instalado en el servidor - Versión GeoServer 2.9.1 - Dispone de herramienta de administración en entorno web (no hace falta instalar nada)
API	OpenLayers	- Dos posibilidades: descargarse la librería y guardarla en el propio servidor o acceder a ella a través de un link. - Versión OpenLayers 2

2.1.2. Datos

En la medida de lo posible la intención es disponer de datos abiertos ('open data') cuyas licencias permitan su utilización sin ningún tipo de restricción (como pueden ser las licencias Creative Commons) y de una fuente fiable y/u oficial, proporcionadas por administraciones públicas. En el caso de la información geográfica es la Directiva Europea INSPIRE (INSPIRE, 2016) la que garantiza la interoperabilidad y la armonización entre los distintos repositorios de datos espaciales. Estos datos deben ser accesibles a través de servicios de red y todas las infraestructuras espaciales serán compatibles entre sí y accesibles a través de un geoportal comunitario, facilitando como mínimo y con carácter, en la mayoría de los casos, gratuito los servicios de localización y visualización de datos espaciales.

En la Comunidad Foral de Navarra, dentro del desarrollo del SITNA, es la Infraestructura de Datos Espaciales de Navarra (IDENA) la que responde a los requisitos definidos por la Directiva INSPIRE (SITNA, 2016a). IDENA, cumpliendo con las características de una IDE, ofrece los siguientes servicios: visualización de los datos a través de servicios Web y, opcionalmente, su consulta; búsqueda y localización de conjuntos de datos y servicios a través del contenido de sus metadatos y descarga de parte o del conjunto de los datos. En este caso se ha optado por descargar los archivos físicamente (Anexo II) y no acceder a ellas mediante servicios Web, ya que lo que se pretende es modificar una copia de los datos para adaptarse a las necesidades del estudio y además tenerlos descargados físicamente proporciona el control de disponibilidad.

En el caso de que en un futuro se quiera trabajar en otros territorios fuera de Navarra el listado de las distintas IDEs a nivel estatal se encuentran en el geoportal de la Infraestructura de Datos Espaciales de España (IDEE), donde se integran en primer lugar todas las IDEs oficiales de datos a nivel tanto estatal como autonómico y local, y en segundo lugar datos de todo tipo de infraestructuras sectoriales y privadas (Ministerio de Fomento, 2016).

Otra fuente de datos espaciales abiertos es el proyecto Open Street Map (OSM). Su objetivo es crear un mapa libre digital, editable y bajo una licencia que protege el proyecto del uso injustificado de terceros. Lo que le distingue de otras fuentes es que los datos los crean y editan voluntarios en una base de datos común accesible por cualquiera: Volunteered Geographical Information, VGI (Haklay, 2010).

Una vez se hayan definido las fuentes de los datos, lo siguiente es determinar qué datos resultan interesantes. El dato básico y fundamental para este tipo de aplicaciones es la mancha de inundación, que será primordial para conocer qué elementos se verán afectados. Como ya se ha expuesto en la introducción, cuando los aforos marcan un cierto caudal se activa un nivel de emergencia distinta. En el caso de Navarra, IDENA proporciona la capa de 'Periodos de retorno' de una inundación[3], que representan las manchas teóricas para cada periodo, que es el dato que utilizará en un principio para cada municipio, ya que la idea es que las manchas se vayan cambiando dependiendo de las experiencias vividas. La cuestión es que hay que intentar relacionar el caudal medido con un nivel de emergencia determinado y qué mancha de periodo de retorno le corresponde a ese nivel.

Cuando ocurra una nueva inundación esas manchas se modificarán según lo ocurrido. Para determinar cuál ha sido el alcance de la inundación existen métodos de teledetección (Elkhrachy, 2015) que mediante sensores ópticos como pueden ser el de Landsat TM o MODIS (Alouene & Petropoulos, 2013; Auynirundronkool et al., 2012; Ogilvie et al., 2015) o RADAR (Mason, Giustarini, Garcia-Pintado, & Cloke, 2014) son capaces de detectar y determinar la mancha que hayan podido dejar. También el uso de los UAVs se está aplicando en este ámbito (Ip et al., 2006; Sohn, Heo, Yoo, Kim, & Cho, 2008) para mejorar tanto en resolución espacial como temporal, aunque se deben tener en cuenta los permisos de vuelo.

Sin embargo estos métodos superan los objetivos de la aplicación y aunque se puedan estudiar estas posibilidades en un futuro, en estos momentos la idea es cambiar y digitalizar las manchas a mano con la

[3] Los periodos de retorno de una inundación hacen referencia a la estimación de la probabilidad de que ocurra una inundación. Es una medida estadística basada en los datos históricos que indican el intervalo medio de recurrencia durante un periodo prolongado de tiempo (Dottori et al., 2016).

aportación y propuestas de los usuarios de la aplicación, es decir, que el encargado de gestionar la inundación en cada municipio tenga la posibilidad de definir lo ocurrido e insertarlo en un histórico para que las manchas teóricas se sustituyan por manchas prácticas relacionadas con la experiencia.

En cuanto a los datos espaciales afectados en inundaciones que deberán aparecer en un 'Plan de Actuación Municipal' y por lo tanto en la aplicación web a desarrollar, serán aquellos que de alguna manera u otra se deben gestionar cuando ocurre una inundación.

Para saber cuáles son estos elementos lo adecuado es conocer qué tipo de actuaciones son los que se describen en el Plan y ver si tienen asociados de alguna manera cualquier información geográfica que resulte útil que aparezca en el visor GIS. Sin embargo, los elementos que se vayan a representar siempre dependerán de la localidad en el que se quiera implantar la solución NOE.

Cada localidad muestra una distribución y urbanismo concreto y será difícil determinar de antemano todos y cada uno de los elementos a tener en cuenta. Aun así sí que existen ciertos elementos que al ser los más comunes será necesario tenerlos en cuenta. Estos pueden ser algunos de los más frecuentes:

- Infraestructuras de transporte
    - Calles
    - Carreteras generales
    - Ferrocarril
    - Aeropuertos
- Puntos de interés
    - Centros de salud
    - Centros educativos
    - Instalaciones deportivas
    - Consistorios
    - Alojamientos turísticos
    - Recursos turísticos
    - Buzones
- Servicios
    - Depuradoras
    - Centrales, subestaciones y centros de transformación eléctrica
- Centros económicos
    - Centros comerciales
    - Polígonos industriales
- ….

Aquellos datos que no existan se tendrán que crear y los ya existentes analizar si es necesario modificar su geometría para ajustar al estudio. Es decir, si al analizar el municipio en concreto, por alguna razón, se decide que algún elemento es mejor crearlo o cambiarlo (no coincide con la mancha y quiero que aparezca, coincide pero no quiero que aparezca…) con el modo de edición del QGIS será posible hacerlo (Anexo III).

Y es en este punto donde hay que decidir si es mejor trabajar con puntos y líneas, o polígonos. Ambas opciones tienen ventajas y desventajas. Para comenzar, en lo que a la eficiencia en el almacenamiento se refiere los puntos y las líneas son más eficaces, menos pesados. En cuanto a la geometría cada uno tiene sus ventajas respecto al otro como se puede apreciar en los siguientes ejemplos (figuras 11 y 12):

Figura 11. Periodo de retorno del río Arga en el campus de Arrosadia de la UPNA-NUP

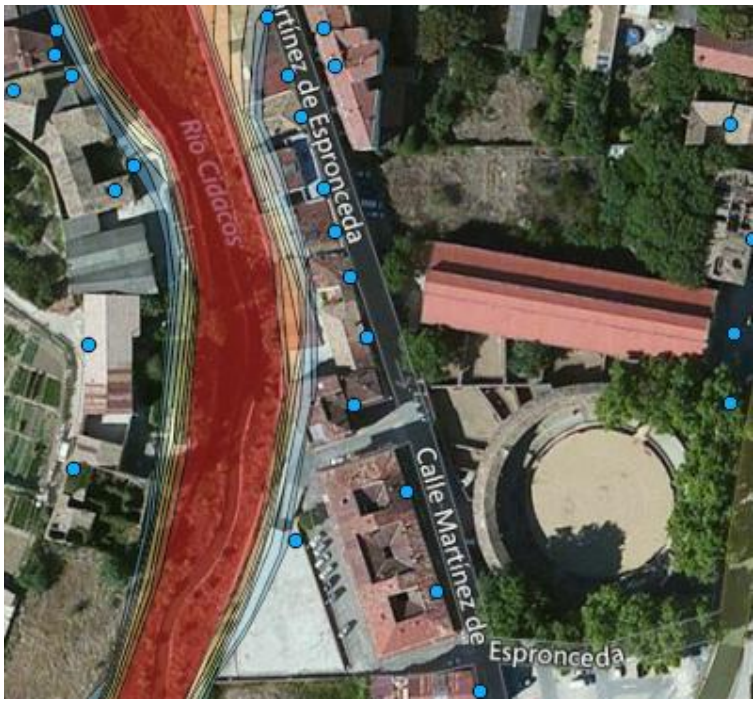


Figura 12. Periodo de retorno del río Cidacos a su paso por Tafalla

Por ejemplo en el caso del campus de la UPNA-NUP de Arrosadia (figura 11), éste se encuentra en un único polígono de catastro, lo que hace que un periodo de retorno de nivel bajo que intersecte en la esquina del polígono incorpore a todo el campus como área de riesgo. En el caso de Tafalla (figura 12), cogiendo los portales (puntos) como elementos representativos, es difícil que coincida la ubicación del portal con la zona afectada del inmueble. En ambos casos la solución pasa por la supervisión y la edición manual.

Dependiendo de la naturaleza del elemento que se quiera representar se decidirá con qué tipo de geometría conviene trabajar. En la metodología se definirá cuál ha sido la decisión tomada en cada momento y por qué.

2.2. METODOLOGÍA

Una vez seleccionada la herramienta a utilizar y la procedencia de los datos que se pretenden implantar en la aplicación web, en los próximos puntos se definirá la metodología seguida para el diseño y desarrollo de la carga de datos geolocalizados de manera que su implantación quede procedimentada en la medida de lo posible sin depender del municipio. Además, se han diseñado e integrado los datos geolocalizados en la aplicación web NOE.

La idea básica de este trabajo (figura 13) consiste en recoger los datos espaciales de las fuentes existentes, incorporarlas a la base de datos, preparar los datos para que estén listos para la publicación, publicarlos y finalmente visualizarlos para su integración en la aplicación web. Además se pretende que tanto la aplicación como el plan que intenta reproducir sean lo más dinámico posible, ya que siempre pueden existir diferencias entre lo teórico y lo que realmente ocurre (cambios en el río, obras realizadas que modifiquen la morfología, fallos en los modelos…). Para conseguirlo la idea es que cada vez que ocurra una inundación, el operario encargado de gestionarlo se base en la experiencia vivida para que tenga la posibilidad de adaptar la mancha de inundación o introducir cualquier elemento que hasta ese momento no se ha tenido en cuenta.

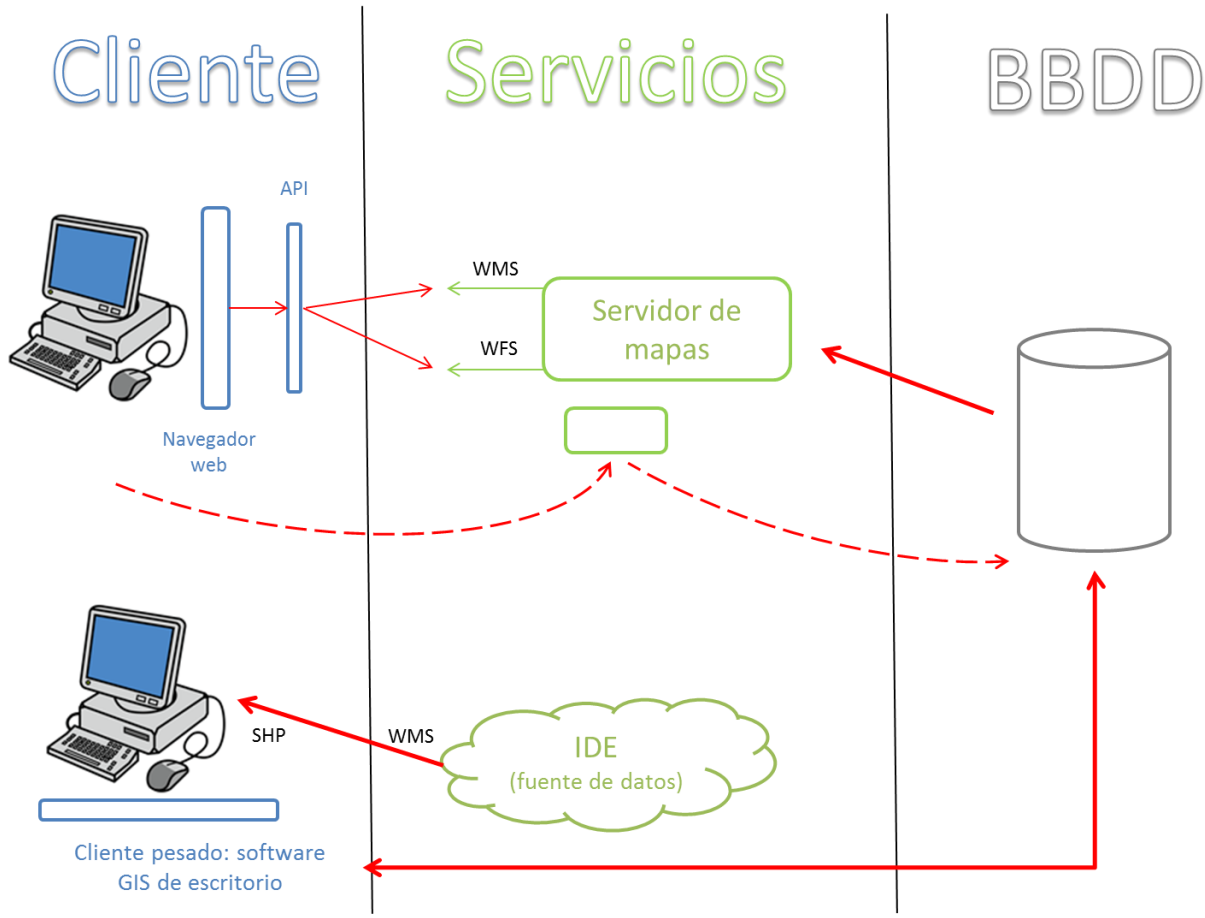


Figura 13. Diagrama de la metodología

Sin embargo, lo cierto es que la mayoría de los municipios todavía no tienen ningún 'Plan de Actuación Municipal' redactado y por lo tanto en esos casos el primer paso consistiría en proponerles uno (para después conseguir la homologación por parte del Departamento de Protección Civil de Navarra). Por eso se puede incluir dentro del flujo de trabajo un análisis GIS para que determine cuáles son los elementos afectados y a partir de estos datos definir las actuaciones, por ejemplo si el agua llega hasta un punto en una calle que el corte de circulación se haga más arriba o añadir a una calle o punto de interés una actuación determinada como avisar a los vecinos afectados.

En el informe se expone mediante algunos ejemplos prácticos qué se debe hacer y por qué, y para conocer cómo hacerlo con las herramientas GIS que se han elegido anteriormente se pueden consultar los anexos. Estos anexos pretenden servir como guías o pequeños tutoriales para poder ejecutar paso a paso cada uno de los procesos, posibilitando realizar el trabajo sin depender del municipio en el que se centre la aplicación NOE.

2.2.1. Pre-procesamiento de los datos

El pre-procesamiento empieza con los archivos Shapefile descargados desde IDENA y consiste en eliminar, unir o modificar las tablas y sus atributos para que se puedan incluir en la base de datos (figura 14). Los datos descargados se podrían dividir entre los que hacen referencia a toda Navarra en su conjunto y los que pertenecen a un municipio en concreto. Para conocer los procedimientos y códigos ejecutables específicos para cada herramienta se puede consultar el Anexo IV.

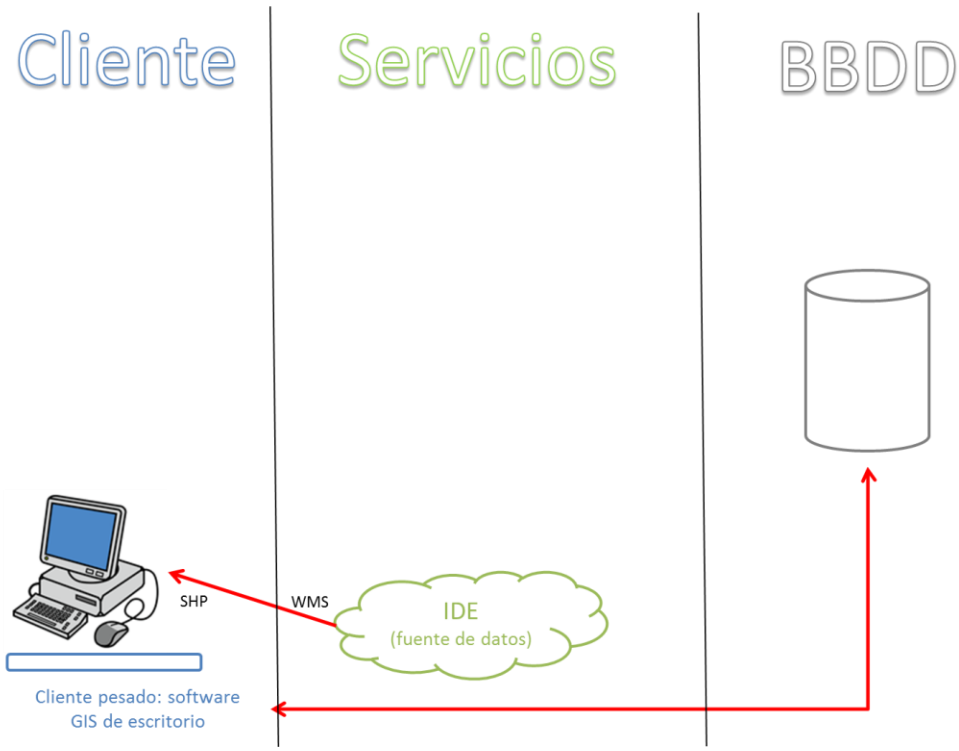


Figura 14. Relación entre las herramientas que se utilizan en el pre-procesamiento

La aplicación web NOE en estos momentos solamente está implantada en el Ayuntamiento de Pamplona/Iruña y se pretende conseguir que otros municipios de Navarra se sumen en un futuro. Por esta razón, y para que el trabajo se automatice en la medida de lo posible, se ha querido evitar que cada vez que se estudie un nuevo municipio se tenga que volver a trabajar con los mismos. Para ello se almacenarán todos los datos relativos a Navarra en una base de datos para que puedan ser consultadas y utilizadas en cualquier momento.

La idea es que cuando se quiera implementar la aplicación en un nuevo municipio baste con que se descargue la información catastral del municipio, ya que el resto estará guardado en la base de datos de forma constante y accesible.

Estos datos generales serán tres:

- Periodos de retorno de la inundación
- Ejes de las carreteras generales
- Puntos de interés: en este grupo entran todos aquellos elementos que por algún motivo interesa conocer si están en peligro o no, como por ejemplo centros de educación o centros sanitarios. Se

representarán con un punto para que sea más fácil cambiar su posición si se decide que el punto coincida con la mancha de inundación (figura 15) y se ha decidido que componga una única capa para que así sea más fácil integrarlos en la aplicación. Si no existe ninguna IDE que proporcione un dato que se considere que va a ser útil se tendrá que digitalizar a mano. Todos los elementos que se vayan a incluir en este grupo deberán constar de los mismos campos, que son: 'tipo' para agrupar los puntos por temáticas, 'nombre' para diferenciar un punto de otro y 'otro' será el campo donde se introducirá la información extra para ese punto.

Figura 15. Ejemplo de un punto de interés desplazado para que coincida con la mancha en Caparroso

Lo más adecuado para conseguir cierta rapidez a la hora de trabajar será que cuando en algún municipio se quiera editar algo (cambiar de sitio como se ha explicado antes o introducir un nuevo elemento que en esa localidad sea de gran interés) se conecte desde el QGIS a la base de datos espacial y hacer los cambias mediante la conexión. De esta manera los datos espaciales almacenados serán siempre los mismos para todos los casos.

En lo que se refiere a los datos específicos para cada municipio se encuentra el catastro. Entre todos los archivos del catastro es el contorno del municipio el que se empleará para seleccionar los elementos que queden dentro del límite y de este modo la cantidad de datos a procesar será menor, ahorrando tiempo en el análisis posterior. Además del contorno, de entre los datos del catastro también se emplearán los ejes de las calles, los portales y las parcelas tanto urbanas como rústicas.

Las parcelas serán de tipo polígono y no se representarán como puntos para que sea posible conocer solamente la porción de parcela afectada utilizando la herramienta de geoprocesamiento adecuada. La intención es que en la aplicación se pueda apreciar hasta qué punto llega la mancha de inundación y que se pueda conocer a qué parcela corresponde, algo así como saber qué trozos de parcelas coinciden con la mancha y no tanto sacar un listado de parcelas afectadas, que puede llevar a confusiones tal y como se ha comentado en la figura 11. La ventaja que tiene el polígono en este caso es que con las parcelas es posible cubrir la mayor parte del territorio y disponer así de un elemento que contenga algún tipo de valor en cualquier localización.

Para tener seleccionados los elementos que pertenecen a un municipio es posible trabajar de dos maneras: creando nuevas tablas o utilizando vistas (*views*). La mayor ventaja de las vistas es que solamente se almacena una copia de la información, lo que implica que, además de ocupar menos, no se tenga problemas de integridad. La integridad conlleva a que un cambio en las capas originales se refleje directamente en el resultado, es decir, que en vez de tratar con tablas estáticas se trabaje con información que va cambiando dependiendo de si se ha modificado algo en las tablas originales.

El inconveniente es que es más complejo en cuanto a estructura acarreando más tiempo en los pasos posteriores. Es decir, si para publicar unos datos se llama a una vista, en ese mismo instante se hace el cálculo de la consulta SQL que compone la vista, y cuanto más costoso sea ese cruce más tiempo necesitará para publicar, llegando en algunos casos hasta no poder publicar.

En las pruebas realizadas para este trabajo se ha visto que la utilización de vistas en la base de datos impide por completo la posterior publicación, y por lo tanto se recomienda trabajar con tablas en lugar de vistas. De este modo, aunque se ocupe más y no haya integridad se asegura que se podrá publicar toda la información. Si de alguna manera se consigue mejorar el rendimiento de las herramientas empleadas (sobre todo el servidor) se recomendaría trabajar con vistas, ya que el control de integridad facilita el trabajo cuando se quiera modificar cualquier elemento de las capas originales descargadas.

Al final lo que queda en la base de datos (figura 16) es una única base de datos con:

- Un esquema para los datos de Navarra: periodos de retorno, ejes de carreteras generales y puntos de interés (modificable).
- Un esquema nuevo para cada municipio interesado en la aplicación. En ella se almacenan por una parte las capas referentes al catastro, que son el contorno, las parcelas (urbanas y rústicas), los portales y los ejes de las calles, y por otra parte las tablas referentes a la base de datos anterior (periodos de retorno, ejes de carreteras y puntos de interés) que han sido creadas cruzando las originales con el contorno municipal mediante consultas espaciales SQL.

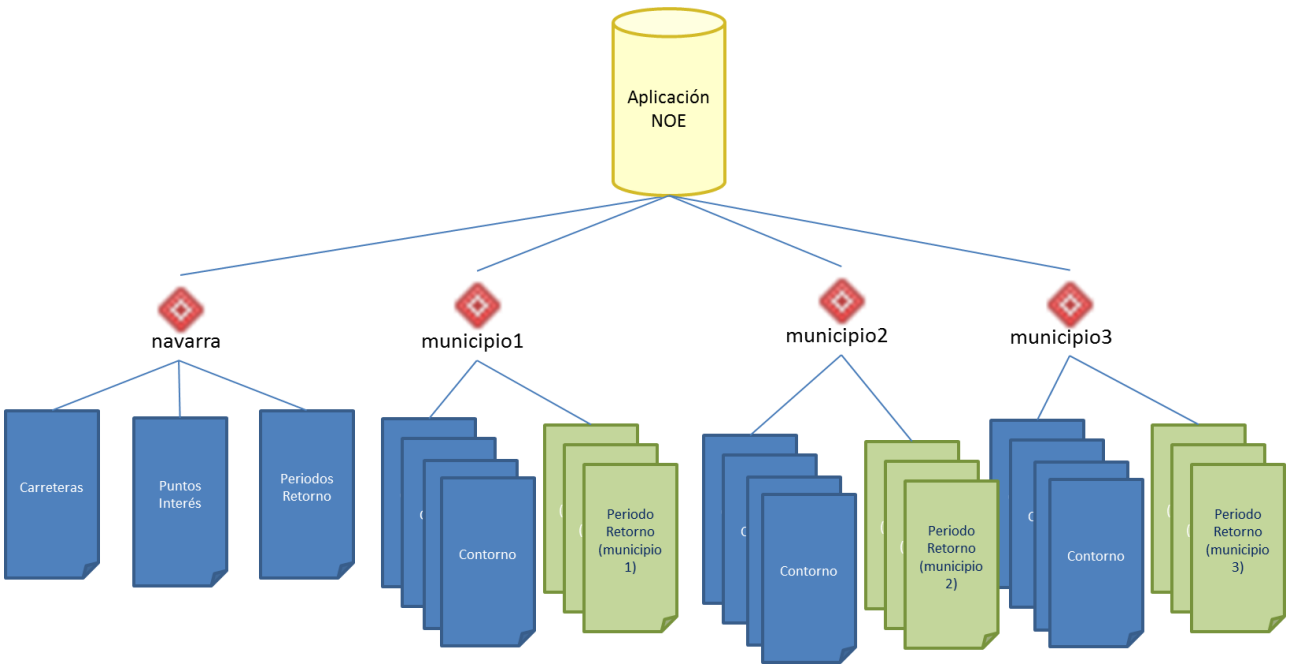


Figura 16. Estructura de la base de datos

En caso de trabajar con vistas todas las modificaciones que se hagan en la geometría, como es el ejemplo de la figura 15, se deben hacer en las tablas y no en las vistas, es decir, en las tablas de color azul de la figura 16. La razón es que no es posible hacerlo en las vistas: QGIS no puede representar las vistas ni mucho menos cambiarlas. Pero aun así esto no provoca ningún error, ya que la garantía de integridad que antes se comentaba ayuda a que cualquier modificación en las tablas originales se propaga automáticamente en todas las vistas posteriores (hasta llegar al visor de la aplicación web final).

También facilita el trabajo posterior eliminar aquellas columnas que se sabe de antemano que no aportan ninguna información adicional o de interés y que solamente aumentan el tamaño de la tabla. Las columnas que deben quedar son al fin y al cabo las que se quieren que muestre la aplicación, por ejemplo en el caso de las calles afectadas sí que aporta información extra en la aplicación el nombre de la calle pero el nombre

del municipio y su código no aportan nada nuevo, el usuario conoce bien a qué municipio hace referencia. Sin embargo este paso se puede dar tanto en el pre-procesamiento como en el geoprocesamiento, la cuestión es que la base de datos, antes de publicar la tabla como capa, solamente contenga los campos finales.

Al importar todos los Shapefiles a la base de datos PostGIS habrá que tener en cuenta que es imprescindible asignarles un sistema de referencia adecuado, que en el caso de la mayoría de los datos espaciales descargados desde el IDENA es el EPSG: 25830. Si no se especifica esta característica será imposible realizar ningún proceso geoespacial, ya que las herramientas que trabajan con datos espaciales necesitan conocer a qué sistema le corresponden las coordenadas que están utilizando.

El sistema de referencia que el European Petroleum Survey Group (EPSG) identifica con el código 25830 se refiere a la ETRS89/UTM zona 30N. El ETRS89 es el datum geodésico ligado a la parte estable de la placa continental europea y el UTM, Universal Transverse Mercator, indica que se trata de un sistema de coordenadas cartesianas proyectado en una proyección cilíndrica conforme. El huso (más o menos se refiere al cilindro) que le corresponde a Navarra es el 30 (figura 17) y en la zona Norte.

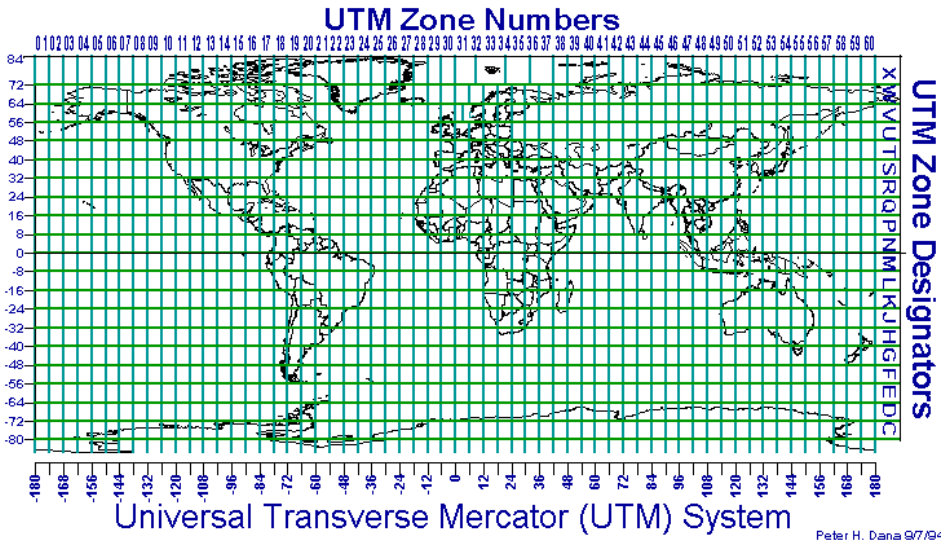


Figura 17. Husos UTM

Con el fin de tener organizado la base de datos, se recomienda que el nombre de todas las tablas que estén almacenadas dentro de un esquema de municipio empiecen con *pre_* (de pre-proceso) para indicar que ya han sido pre-procesadas.

2.2.2. Geoprocesamiento

El geoprocesamiento empieza con todos los datos espaciales iniciales almacenados en la base de datos preparados para que sean utilizados. Y el objetivo es conocer qué elementos se ven afectados en cada periodo de retorno. Es en este apartado donde se saca la información geográfica para elaborar el plan y se crean las capas finales que se van a mostrar en el visor de la aplicación web (algunas coincidirán pero otras no). A la hora de visualizar estas capas en la aplicación web será inevitable conocer el periodo de retorno al que pertenecen para poder mostrar así solamente los datos referentes a la emergencia que se está simulando.

Se recomienda que los nombres de aquellas capas que serán publicados posteriormente empiecen con un *publicar_* para que la base de datos se ordene y así poder identificar con más rapidez las capas que se quieran representar, mientras que las tablas que solamente se van a obtener para ayudar a la redacción del plan empiecen con un *plan_*.

Aquellas tablas que se implementen en el visor, además del periodo de retorno que les afecta, también necesitarán campos que se quieran mostrar mediante pop-ups, como por ejemplo el nombre de una calle o tipo de un punto de interés.

La manera de trabajar consiste en la interacción directa y continua entre el software GIS de escritorio y la base de datos espacial (figura 18): el software GIS de escritorio se conecta a la base de datos, recoge la información, modifica o crea capas y los vuelve a guardar de nuevo en la base de datos. Sin embargo siempre cabe la posibilidad de aplicar algoritmos de geoprocesamiento mediante consultas SQL directamente en PostGIS. Para conocer los procedimientos y códigos ejecutables específicos para cada herramienta se puede consultar el Anexo V.

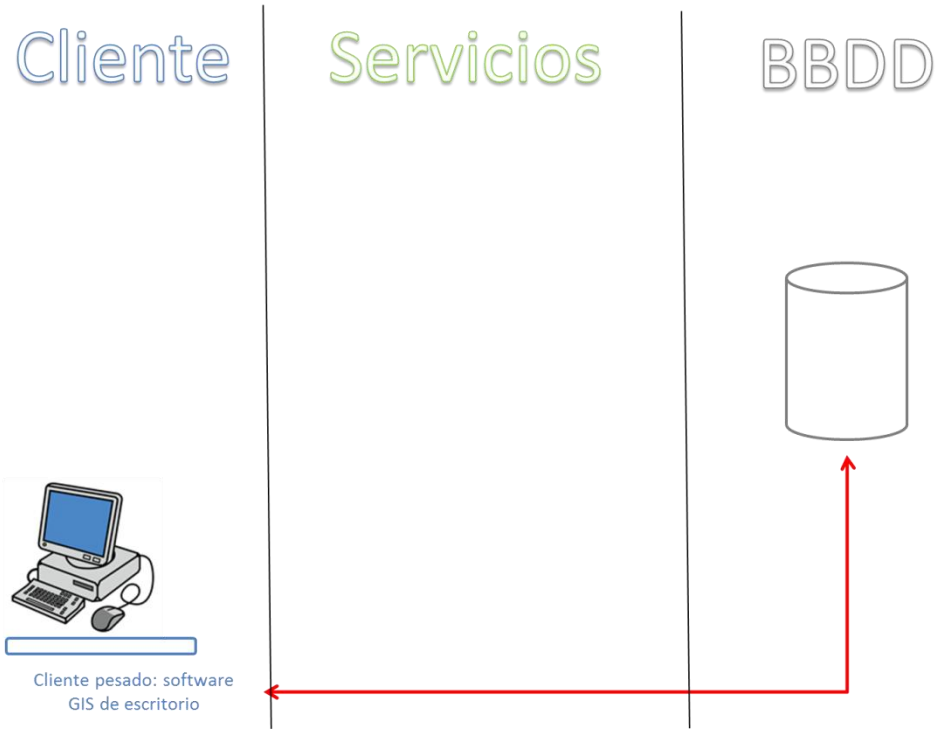


Figura 18. Relación entre las herramientas utilizadas en el geoprocesamiento

Crear un modelo en QGIS ayuda a automatizar el proceso mediante el encadenado de todos los geoprocesos a ejecutar. De este modo, además de agilizar el proceso y facilitar el trabajo, se evita tener que ir guardando todos los archivos que se van creando en cada proceso, permitiendo trabajar más limpio y ordenado. Aun así la mayor parte de los geoalgoritmos existen tanto en el software GIS de escritorio como en la base de datos espacial.

En los siguientes pasos se define qué se ha hecho con cada capa de la base de datos y por qué. Se podría considerar que cada tabla almacenada en la base de datos necesita un geoprocesamiento distinto, por lo tanto este apartado se ha dividido en tantas partes como tablas.

2.2.2.1. Periodos de retorno

Es la capa básica para este tipo de trabajos debido a que el trabajo consiste en conocer cuáles son los elementos de otras capas que queden dentro de la misma, y para ello habrá que ir comparando cada capa con la mancha de inundación de los periodos de retorno.

El único problema que se ha observado en esta capa es su geometría, ya que dependiendo del municipio es más simplificada o más compleja (figura19): en el caso de algunos municipios, como por ejemplo Tafalla, las manchas están dibujadas con polígonos más simples y además sin tener en cuenta los edificios (a partir de un MDT); sin embargo, en otros municipios como es el caso de Funes, se nota cómo la capa ha sido vectorizada a partir de un ráster, dibujando cada contorno de los píxeles, y respetando los edificios (se

observa que la mancha solamente cubre las calles por donde pasa, creando pequeñas 'islas' en algunos de los casos, es el efecto que provoca la automatización a partir de una orto).

Figura 19. Manchas de inundación en Tafalla y Funes

No cabe duda de que la representación del segundo caso sea más detallada y el que mejor se ajusta a la realidad, pero para la aplicación que se está desarrollando ésta geometrías pueden acarrear ciertos problemas:

- Cuando se quieran cruzar manchas con parcelas urbanas, la forma del contorno condiciona si esa parcela se considera afectada o no. Es decir, en el ejemplo de la figura 20 todas las parcelas de esa calle deberían de constar todas juntas como parcelas afectadas o no afectadas, pero en este caso las parcelas 1, 2 y 4 sí que se incorporarían dentro de las afectadas porque la esquina de algún píxel cae dentro mientras que la parcela 3 no se incorporaría porque ningún píxel cae dentro.

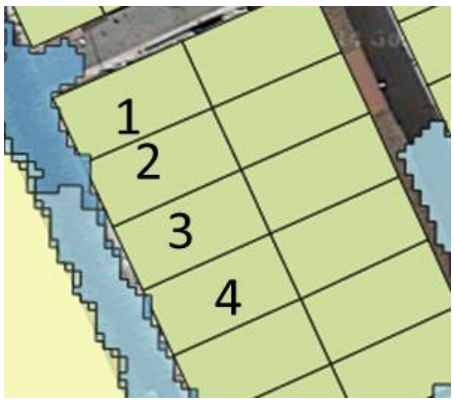


Figura 20. Ejemplo de intersección entre mancha y parcelas urbanas

- En el caso de ciertos algoritmos el hecho de que la capa contenga tantas líneas (cada borde de los píxeles) conlleva un coste computacional muy elevado, ya que el cálculo se lleva a cabo con todas las líneas una a una. La diferencia en cuanto al tiempo de ejecución

La solución pasa por simplificar la geometría (figura 21), pero solamente en el caso de aquellos municipios que tengan este problema. Con una tolerancia adecuada la cantidad de líneas se reduce significativamente evitando la mayoría de los problemas. Sin embargo todavía pueden quedar algunos casos donde se continúe dejando fuera algunas parcelas. Para esos casos es posible añadir un 'Buffer' a la mancha o editarlas a mano (consultar el anexo III). También se pueden analizar las islas y ver si se eliminan o interesa dejarlos como están.

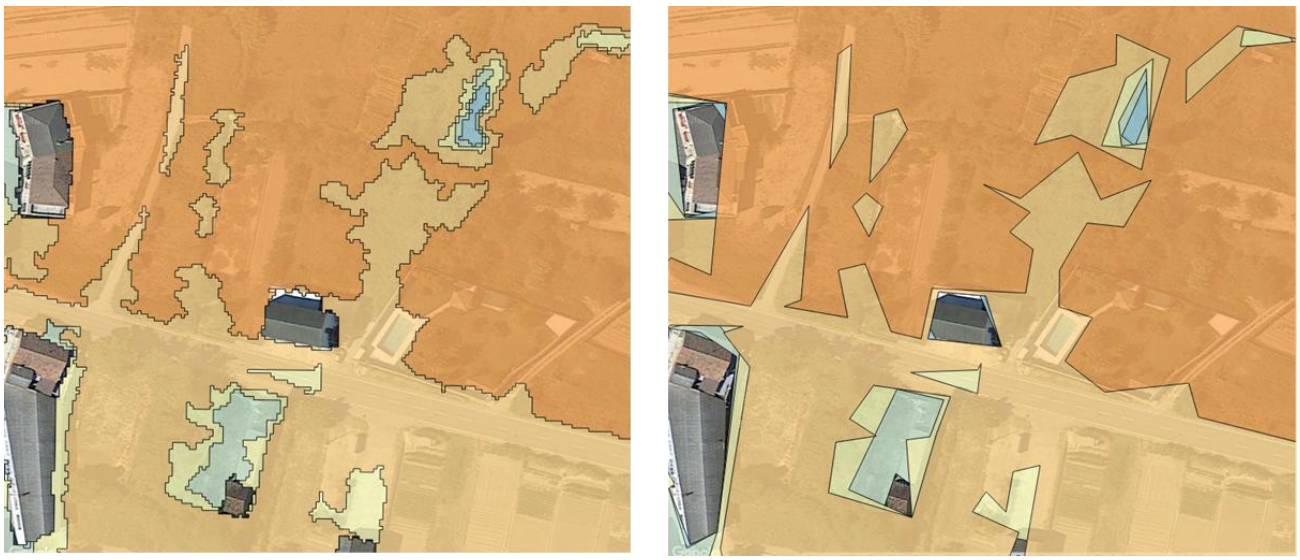


Figura 21. Simplificación de la geometría con una tolerancia de 5

2.2.2.2. Calles y carreteras

Debido a que las calles se descargan junto al catastro y las carreteras generales se encuentran dentro de la temática de infraestructuras, en la base de datos aparecen como dos capas distintas cada una con sus propios campos. Pero para este trabajo en concreto el procedimiento con ambas será similar, ya que se trata en ambos casos de elementos lineales que se quiere conocer el punto de corte con la inundación.

El objetivo con las calles y carreteras es doble: para comenzar se quiere conocer el listado de las calles o carreteras afectadas en cada periodo de retorno y además calcular la localización exacta del punto de intersección. Las calles afectadas podrán ser utilizadas tanto para elaborar el plan como para mostrarlas en la aplicación, pero los puntos de intersección entre calles y manchas no se representarán como tal en la aplicación, si no que sirven para poder determinar manualmente en el plan de actuaciones el punto exacto para cortar la circulación (figura 22), que sí se mostrará en el visor web.

Figura 22. Diferencia entre punto de intersección y punto de corte de circulación (información vs actuación) en Tafalla

Las capas descargadas desde el IDENA representan las calles/carreteras por tramos, es decir, las calles/carreteras están compuestas por más de una línea. Para la aplicación que se está desarrollando esto conlleva a que a la hora de querer sacar una única lista de las calles afectadas en cada periodo de retorno el nombre de la misma calle salga repetida más de una vez, dificultado la comprensión del significado de esa lista. Por ello el primer paso en el geoprocesamiento de estos datos es unir todas las líneas que corresponden a una misma calle, lo que se conoce como disolver. Es necesario identificar un campo único que ayude a disolver estas líneas en cada capa.

Una vez se han disuelto las líneas cada uno de los objetivos sigue un proceso distinto.

En el caso de los puntos de intersección entre la mancha y las calles/carreteras el geoproceso es bastante sencillo. Los puntos de intersección solamente se pueden calcular a partir de dos capas lineales. En este caso los periodos de retorno son polígonos, así que primero se transforman a líneas y posteriormente se calcula la intersección entre ambas capas. El resultado es una nueva capa con todos los puntos de intersección que servirá como información para la realización del plan.

En cuanto a la lista de las calles/carreteras afectadas, el geoproceso a seguir es distinto. Al final consiste en encontrar las calles que espacialmente se encuentren total o parcialmente dentro de alguna de las manchas de retorno. Con el geoalgoritmo adecuado será posible encontrar la intersección de ambas capas. El resultado será una nueva capa con las calles/carreteras que quedan dentro de los límites de la mancha y cada elemento tendrá asignada un periodo de retorno determinado, la misma calle aparecerá tantas veces como manchas que lo cortan.Lo siguiente es unir todas las tablas en una y eliminar las columnas que no se quieran incluir siguiendo los criterios marcados anteriormente.

2.2.2.3. Puntos de interés

El trabajo más difícil o costoso con estos elementos ya se ha llevado a cabo en el pre-proceso, ya que mediante la unión de tablas se ha conseguido unificarlos y con la edición en QGIS ubicarlos en una posición ideal para que coincidan con la mancha de inundación si se observa que lo tienen que hacer. Esta capa creada valdrá tanto para elaborar el plan como para representarla en el visor de la propia aplicación para que los usuarios la puedan consultar.

El único geoproceso que deben soportar estos puntos tiene que ver con asignarles en qué periodo de retorno se verán afectados. En realidad, lo que se pretende es crear una tabla donde aparecerán todos los puntos afectados junto al valor del periodo de retorno que les afecta, es decir, un punto aparecerá en esa tabla tantas veces como manchas con las que coincide.

De este modo, al igual que pasa con el resto de capas a mostrar en el visor, es posible mostrar en cada momento solamente aquellos elementos que cumplan la condición de pertenecer a un cierto periodo de retorno.

El geoproceso es el mismo que se ha utilizado para conocer qué datos de Navarra pertenecen a un municipio en concreto y para sacar la lista de calles/carreteras: cortar la capa de los puntos con cada mancha de periodo de retorno. Este proceso será posible hacerlo directamente desde la propia base de datos espacial con consultas SQL, pero también cabe la posibilidad de hacerlo desde el software GIS de escritorio (más costoso en tiempo).

2.2.2.4. Parcelas urbanas y rústicas

Al igual que pasaba con las carreteras generales y las calles o los puntos de interés, las parcelas afectadas podrán ser utilizadas para sacar un listado que se pueda incluir en el plan y determinar así las actuaciones y por otro lado implementarlos en el visor de la aplicación web para aportar información extra para la toma de decisiones.

En este caso, como ya se ha explicado en el apartado 2.2.1 sobre el pre-procesamiento de los datos, las capas escogidas son poligonales para poder conocer qué porción exacto de la parcela es la que realmente está afectada y no asignar como parcela afectada innecesariamente parcelas enteras donde en realidad el

impacto es menor (ver ejemplo de la figura 11, donde todo el campus Arrosadia de la UPNA-NUP se introduce dentro de las parcelas afectadas debido a una superposición mínima en el jardín).

Otra de las ventajas de usar polígonos es que de este modo la mayor parte del territorio queda cubierto, lo que permite extraer información de casi todo los puntos en el mapa, y se podrá decidir si en la aplicación interesa mostrar los datos asociados mediante pop-ups o cualquier otro método.

Lo que ocurre con estos datos es que no tienen asociados ningún tipo de campo que ayude a identificar de forma clara a qué parcela hacen referencia. Por ese motivo resulta interesante tener a mano la capa de los portales, que permiten asociar una parcela a un portal, que estos sí contienen campos más intuitivos como el nombre de calle y el número del portal.

Entonces, lo primero es unir las tablas de parcelas urbanas con las de los portales mediante un campo en común, en este caso la referencia catastral ('REFCAT'). Las columnas restantes podrán ser eliminadas para aligerar la base de datos. A partir de este instante todos los resultados que se van a ir obteniendo después de cada geoproceso que sufra la capa de las parcelas urbanas tendrá asociada el nombre de la calle y el número de portal.

En el caso de las parcelas rústicas no existe ninguna manera de hacer que la información asociada sea más clara para su identificación, y por lo tanto no es posible realizar ninguna unión con otra tabla.

Lo siguiente ya es calcular la intersección de las parcelas con las manchas de inundación, que con el algoritmo adecuado el resultado serán polígonos de las parcelas que quedan dentro de la mancha y parcelas cortadas hasta donde llegan las manchas de inundación (figura 23) con una única tabla donde cada polígono contenga la información de ambas capas originales: nombre de la calle, portal y periodo de retorno. Esta intersección será necesaria hacerla tantas veces como manchas de inundación y después unirlas todas al final, tal y como ocurre con las capas lineales (calles y carreteras) y puntuales (puntos de interés).

Figura 23. De fondo los polígonos de las parcelas urbanas y superpuestas las porciones afectadas por una determinada mancha de inundación

2.2.2.5. Capa de actuaciones

Al fin y al cabo el objetivo de la aplicación web NOE es plasmar el Plan de actuaciones de un municipio dentro de un entorno web donde sea más fácil seguir las instrucciones. Pero no todas las actuaciones tienen que ver con elementos concretos (como puede ser el ejemplo del punto de corte de la circulación), y para esas actuaciones que de alguna manera contienen una componente espacial pero no se pueden calcular mediante geoprocesos se necesita disponer de alguna capa de referencia.

Como se ha decidido que lo mejor para representar estas actuaciones es utilizar puntos, para no crear una tabla más lo mejor es añadir estas actuaciones a la tabla de 'Puntos de interés'. Estos nuevos puntos tendrán que ser digitalizados a manos mediante QGIS siguiendo los pasos expuestos en el anexo III. Sin embargo estos nuevos elementos digitalizados deberán adaptarse a los campos que contiene la tabla de los puntos de interés: 'tipo', 'nombre' y 'otro'.

La manera de rellenar estos campos será introduciendo la palabra 'Actuacion' en el campo 'tipo' y rellenar los otros dos según se vea oportuno dependiendo de la naturaleza de la actuación.

Además se le añadirá una nueva columna denominada 'actuacion' a la tabla, de tipo texto con una longitud de 10. De esta manera se podrá rellenar a mano este campo de todos los elementos para definir un identificador de una actuación en concreto que permita relacionar el punto con una actuación del Plan en concreto, por ejemplo:

- act0: llamar a director de la escuela infantil para evacuar los edificios. El identificador 'act0' se le asigna al punto que representa la escuela.
- act1: avisar a Correos de recoger cartas del buzón. El identificador 'act1' se le asigna al punto que representa el buzón.
- act2: cortar carretera en calle Zubiaran. El identificador 'act2' se le asigna al punto que representa el corte de calle.
- **…**

Esta columna será utilizada para relacionar en la aplicación web los elementos del visor con las actuaciones descritas en la parte izquierda.

2.2.3. Publicación de las capas

La publicación consiste en servir datos espaciales siguiendo especificaciones de la Open Geoespatial Consortium (OGC) para la interoperabilidad dentro de los GIS y la World Wide Web (WWW) para facilitar el intercambio de la información geográfica. El objetivo es publicar con los servicios web de mapas las capas que se han ido creando y que se han almacenado en la base de datos espacial PostGIS para que después la aplicación, mediante la API de OpenLayers en este caso pero puede ser cualquier otro, pueda mostrar los datos dentro de una aplicación web.

En este caso las herramientas GIS que participan en la publicación (figura 24) solamente son dos: la base de datos (PostGIS) y el servidor de datos (GeoServer). Además la comunicación solamente tiene una única dirección, ya que los datos que están almacenados en la base de datos espacial son recogidos por el servidor para que sean publicados siguiendo las especificaciones. Para conocer los procedimientos y códigos ejecutables específicos para cada herramienta se puede consultar el Anexo VI.

Figura 24. Herramientas necesarias para la publicación de datos espaciales

Al estar basado en las especificaciones de la OGC se asegura que cualquier herramienta que vaya a utilizar datos espaciales mediante la web tenga la capacidad para trabajar con ellos. De este modo se garantiza la interoperabilidad geoespacial facilitando la comunicación entre sistemas, ya que los servidores de datos geoespaciales hacen que los datos espaciales estén disponibles en un formato adecuado para la web (Song et al., 2008).

Dentro de estas especificaciones se encuentran distintos tipos de estándares. Para las aplicaciones web las más importantes son las que tienen que ver con los servicios web, que son los que realmente agrupan los protocolos y estándares y permiten intercambiar datos entre aplicaciones web. Los más importantes son el Web Map Service (WMS) y el Web Feature Service (WFS).

EL WMS es un servicio de visualización que permite visualizar datos vectoriales y ráster permitiendo operaciones sencillas mientras que el WFS es un servicio de descarga que ofrece la posibilidad de acceder en la web a todos los atributos de fenómenos individuales de un conjunto de datos para poder efectuar análisis complejos (Steiniger & Hay, 2009).

En el caso de esta aplicación lo que realmente interesa es servir capas como WFS para que la aplicación pueda ejecutar acciones más complejas sobre las capas. Aunque sí se utilicen capas WMS para ponerlas como mapas base, estas no serán publicadas desde el servidor de la aplicación, si no que se va a acceder a una fuente externa.

El hecho de disponer de capas WFS facilitará la labor de diseñar una aplicación web dinámica e intuitiva, debido a que será más fácil acceder a cada uno de los elementos y sus correspondientes atributos. Entre otras acciones se podrán ejecutar con menos dificultad:

- Mostrar solamente aquellos datos que cumplan una determinada condición, en este caso que muestre la mancha de inundación, elementos afectados y actuaciones que estén vinculadas a un determinado nivel de emergencia. Con tener publicada solamente una tabla que contenga una columna con el periodo de retorno/nivel de alerta, será más fácil seleccionar solamente aquellos elementos que interesen.
- Al poder tratar cada elemento independientemente será más fácil asignar estilos a cada elemento sin tener que preparar los estilos en la propia publicación, y además favorece el poder insertar pop-ups a la aplicación donde se muestren los campos de forma más amigable que un GetFeatureInfo de un WMS o la posibilidad de vincular un elemento del mapa con otro elemento exterior (por ejemplo para que cambien de color a la vez las actuaciones del listado y su correspondiente punto en el mapa).

Otra de las funcionalidades que tiene la conexión entre PostGIS y Geoserver es que esta comunicación es dinámica: cada vez que se modifique una tabla en la base de datos, si esta ha sido publicada por el servidor de mapas anteriormente, automáticamente el servicio web proporciona desde ese mismo instante la capa ya modificada.

Una ventaja de esta funcionalidad es que cualquier cambio que se haga en la base de datos repercute directamente en la aplicación web final, sin tener que modificar ninguno de los pasos que se sigue. La conexión entre las herramientas permite la automatización en la carga de datos.

Pero uno de los inconvenientes de tener que publicar cualquier tabla una única vez es que se tenga que tener en cuenta que es posible que sufra modificaciones. Es decir, a la hora de publicar una capa, una de las opciones a rellenar es la 'Bounding Box' (bbox). Si se decide que la extensión de la bbox de la capa publicada coincida con la bbox de la capa original, puede que cualquier cambio que se haga en el futuro no se pueda visualizar porque se haya superado los límites espaciales (el elemento sobresale del bbox de la capa publicada).

La solución puede ser poner a todas las capas la extensión del municipio y así asegurar que todos los elementos se podrán visualizar, o cada vez que se haga una modificación de la capa original comprobar que este esté dentro de los límites y en caso contrario actualizar la bbox de la publicación.

El flujo de trabajo para la publicación de capas en GeoServer se divide en tres etapas fundamentales: crear un espacio de trabajo, crear un almacén de datos y publicar capas.

Los espacios de trabajo se puede decir que se crean para crear un proyecto dentro de GeoServer, contenedores para agrupar capas similares. Tener el servidor divido en espacios de trabajo (o proyectos) ayudará a organizar el servidor y gestionarlo con más eficacia. En este caso es suficiente con crear un único espacio de trabajo para ésta aplicación web (con el nombre de 'AplicacionNOE') e ir creando más espacios si en un futuro surge otra aplicación web GIS en la empresa. Los servicios web que se le asignarán a este espacio de trabajo son el WMS y el WFS.

Lo siguiente es crear un almacén de datos asociado a un único espacio de trabajo, que es el contenedor de datos geográficos que puede estar conectado a una fuente exterior de datos ráster o vectorial. Estas fuentes de datos podrán ser de diferentes tipos, como directorios de Shapefiles, PostGIS, WFS, WMS en cascada, GeoTIFF, ImageMosaic…

La aplicación web del proyecto ya se ha organizado para que los datos se almacenen en una base de datos espacial PostGIS, así que esta será la fuente de los datos. A la hora de crear un almacén de datos solamente permite crear uno por cada esquema, que en este caso significa que se creará un almacén por municipio.

Lo último será publicar las capas. Las opciones a rellenar son entre otros el nombre, título, resumen, sistema de referencia, 'bounding box', estilo para el WMS o campos a mostrar. El estilo podrá ser modificado pero en este caso la aplicación consumirá la capa por WFS y el estilo se le asignará en el propio código ejecutable de la aplicación tal y como se ha explicado antes, así que no hace falta definir estilos.

Cuando ya se haya publicado la capa será posible acceder a ella a través de la web si se conoce la URL del servidor. Una de las maneras para conocerla es pre-visualizando la capa con OpenLayers (opción que proporciona el propio GeoServer) y coger toda la dirección hasta la '?', que tendrá este aspecto:

https://servidor/geoserver/espacio_trabajo/wms_wfs

El nombre de la capa para poder utilizarla en la API, tal y como se verá en el siguiente apartado, también se puede consultar en la sección de 'Previsualización de capas':

Espacio_trabajo:**nombre_capa**

Ya que el nombre del espacio de trabajo va a ser la misma en todos los casos, los nombres de las capas deben ser lo suficientemente claros para no duplicar nombres. Por este motivo se recomienda nombrar las capas con el nombre del municipio primero y después un nombre descriptivo de la capa, por ejemplo tafalla_calles, pamplona_calles, tafalla_parcelas_u…

2.2.4. Visualización e integración en la aplicación web

Con toda la arquitectura montada y con el servidor publicando todas las capas necesarias toca integrar los datos en la aplicación web. Ésta es la parte final del tratamiento de datos geográficos afectados en planes de inundaciones: mostrar en la propia aplicación web NOE los elementos espaciales que se han ido creando de manera que coincida la información ofrecida hasta ahora por la aplicación con lo que se representa en el visor.

La aplicación web NOE ya está funcionando con una base de datos donde se almacenan las actuaciones, así que la idea es tener mentalmente separados lo que viene a ser el gestor de las emergencias por un lado (el que está pendiente de si se ha iniciado un Nivel de Alerta, cargar las actuaciones y gestionar el estado de los mismos) y por el otro el visor que se pretende crear con las capas publicadas. Ambas partes se relacionan entre sí a través del identificador de las actuaciones, que en las capas se encuentra en el campo 'actuacion' de la capa de los puntos de interés.

Como ya se ha comentado en la introducción de este trabajo, la aplicación web NOE muestra dependiendo del nivel de alerta un listado de actuaciones definidos en el plan. Estas actuaciones tendrán un estado que el usuario podrá cambiar: por realizar (rojo), en progreso (naranja) y realizado (verde).

El objetivo es que cuando se active un nivel de emergencia se muestre en el visor solamente los elementos afectados correspondientes a ese nivel y su color dependa del estado de actuación con la que esté asociada.

Además el usuario deberá tener la opción de modificar aquellos elementos que considere oportuno. El objetivo es que el propio usuario de la aplicación pueda proponer cómo modificar los datos del plan teórico para adaptarlos mediante datos basados en la práctica. La aplicación debe permitir modificar, borrar o crear nuevos elementos con sus actuaciones asociadas que cambien lo que en un principio se ha creado por uno nuevo que garantice que la próxima vez que se alcance ese mismo nivel de emergencia los datos y actuaciones se ajusten a la emergencia real.

Para lograr todos los objetivos las herramientas (figura 25) son básicamente: el servidor de mapas y la API para JavaScript. Como ya se ha visto en el punto anterior el servidor publica las capas para que otras herramientas GIS las utilicen a través de la red. Para conocer los procedimientos y códigos ejecutables específicos para cada herramienta se puede consultar el Anexo VII.

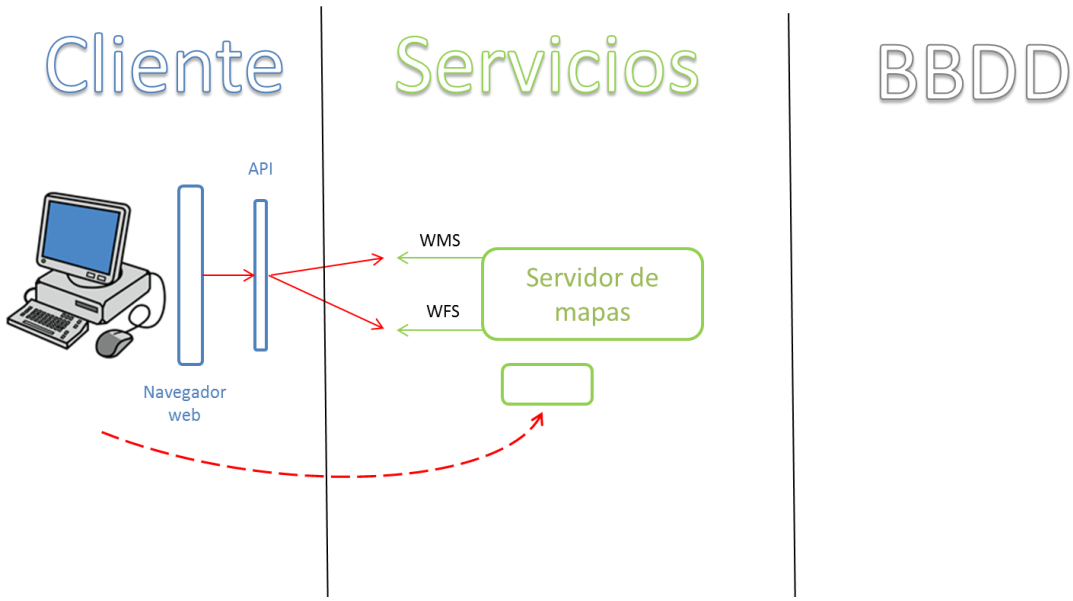


Figura 25. Herramientas necesarias para la visualización de los datos integrados en la aplicación web y su relación

Esta API, apoyándose en las subrutinas, funciones y procedimientos que lo componen, permitirá integrar en la aplicación web las capas servidas y añadir funcionalidades parecidas a un GIS de escritorio para que el resultado final sea lo más amigable e interactivo posible.

Cuando se habla de amigable se refiere a un visor GIS donde no se saturen los elementos y los controles y la visualización sea clara, así como que las funciones de las que disponga sean las suficientes para que la aplicación sea práctica y fácil de entender. La interactividad ofrece por un lado respecto a los demás elementos de la propia aplicación, pero también con los propios datos básicos (los almacenados en la base de datos).

Las propiedades y características del código ejecutable para que un navegador web sea capaz de interpretarlo se definen en los siguientes puntos. Será este el código ejecutable que se envía cada vez que se quiera visualizar el visor de la aplicación web.

2.2.4.1. Estructura del código

El HTML (*HyperText Markup Language*) es el lenguaje de marcado para la elaboración de páginas web, el estándar de referencia definido por el *World Wide Web Consortium* (W3C) que será interpretado por el navegador web para transformarlo en una página web. Se trata de un código de texto plano compuesto por etiquetas que definen el contenido de la página web.

El flujo de mensajes entre el cliente y el servidor comienza cuando el cliente hace una petición de un HTML a un servidor, el servidor crea el archivo HTML que es enviado al cliente, y finalmente el navegador interpreta y renderiza el resultado, que se muestra al cliente. Toda la comunicación se realiza mediante protocolo HTTP (*HyperText Transfer Protocol*) que define reglas y procedimientos de comunicación entre clientes y servidores.

Para diseñar una página web dinámica que reaccione según las acciones del usuario habrá que incorporar un HTML dinámico, en este caso será del lado del cliente mediante código JavaScript, que son scripts insertados en el propio código HTML que se ejecutarán por el navegador web añadiendo dinamismo y funcionalidad a la página web evitando tener que realizar muchas llamadas al servidor. También necesita una hoja de estilos en cascada (CSS) para definir y crear la presentación del documento ya estructurado en HTML.

Partiendo de esta base, el código HTML que se integrará en la aplicación web NOE se dividirá en tantos archivos como lenguajes (HTML, JavaScript y CSS), tanto por claridad comopara facilitar el mantenimiento. En el encabezado del propio HTML se ejecutará el enlace a cada uno de los archivos.

En el encabezado también se deben incluir los enlaces para hacer las llamadas oportunas a todas aquellas librerías que son necesarias para que el código ejecutable funcione como definen las APIs. En este caso será imprescindible introducir un enlace a la API para JavaScript de OpenLayers, que se podrá elegir entre hacerlo al enlace en la web o en local (cargando la librería completa al servidor donde se vaya a cargar el código HTML).

Dependiendo de la capa base que se pretenda incluir (apartado 2.2.4.3) es posible que también sea necesario incluir otro enlace a otra API en el encabezado, como pasa por ejemplo con la API para JavaScript de Google Maps para insertar cualquiera de sus capas base disponibles: callejero, imagen de satélite, híbrido y relieve.

La API de OpenLayers está escrita para JavaScript y por lo tanto todo lo referido al mapa que se pretende representar se encuentra en este archivo. El HTML solamente se encarga de la estructura, define todos los elementos que aparecen en la aplicación web. Todos los demás archivos y las funciones que los componen hacen referencia a los elementos creados aquí.

2.2.4.2. Mapa

La página web se dividirá en dos grandes partes: por un lado el formulario donde aparecerán las actuaciones del Plan y por el otro el propio visor que será un mapa. Este mapa deberá cumplir ciertas características u opciones, que pueden ser por ejemplo:

- Un zoom máximo para que el mapa no se visualice más allá de su resolución. En el caso del mapa base de IDENA también para que el mapa se visualice a un zoom mayor que 15.
- La extensión máxima del mapa que se adecúe al municipio en concreto y que no permita salir de un cierto 'bounding box'. Las coordenadas de este recuadro dependen del sistema de referencia del mapa, que está unido a la capa base seleccionada.
- Un centro de la imagen para que el visor se centre en el municipio. Necesita las coordenadas concretas del centro (depende del sistema de referencia del mapa) y un nivel de zoom inicial.

Puede resultar de ayuda cargar las capas en el software GIS de escritorio para calcular cuáles pueden ser las coordenadas adecuadas para cada municipio en el sistema de referencia que se desea.

2.2.4.3. Capa base

La capa base será aquella que se mostrará siempre como fondo. Se trata de un mapa base que se emplea básicamente para que sirva de referencia, un mapa donde el usuario se puede ubicar y entender dónde se encuentra cada elemento de su localidad. De este modo todas las demás capas que se muestran se superpondrán a esta y ayudará a entender la representación en su totalidad. En general existen dos tipos de capa base: ortoimagen o de origen vectorial. También existen las híbridas, que son la mezcla de ambas (figura 26).

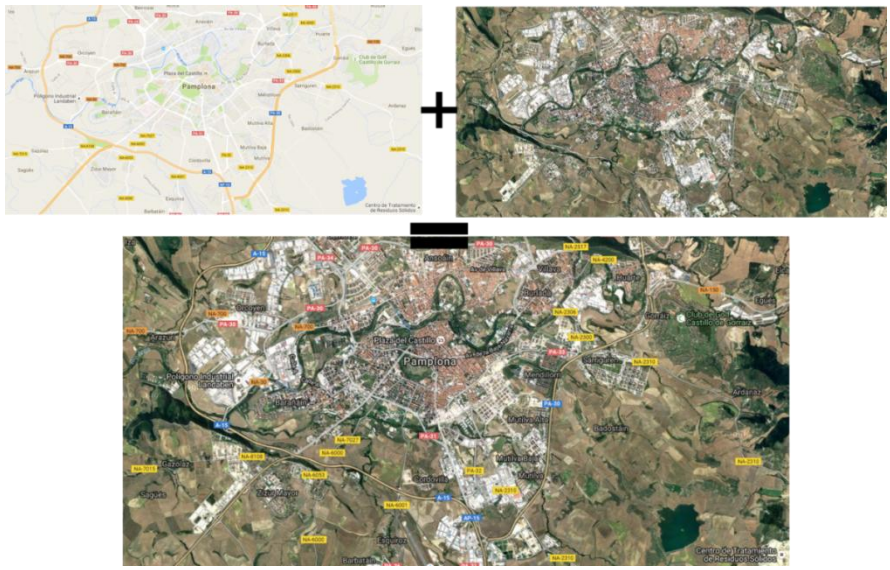


Figura 26. Mapa base híbrida de Google Maps, creada a partir de Google Streets y Google Satellite

La ventaja que tiene una ortoimagen es que las imágenes aéreas proporcionan una información fácil de interpretar por el usuario, ya que la similitud entre la realidad y la fotografía es muy alta. De este modo el objetivo principal de la capa base que consiste en que permita ubicarse de forma rápida dentro de la representación se cumple, siempre y cuando se conozca la zona. Sin embargo, en algunos casos la baja resolución espacial de las ortoimágenes provoca una dificultad extra a la hora de identificar elementos espaciales.

Por su parte, los mapas base de origen vectorial son una representación gráfica pero esquemática de la realidad que mediante simbología pretende representar en dos dimensiones la realidad. La mayor ventaja es que permite representar elementos espaciales que en una imagen aérea no aparecen, y además existe la posibilidad de añadir etiquetas que describen los elementos.

Las híbridas combinan las ventajas de ambas: visualmente son fáciles de interpretar y las etiquetas permiten identificar elementos que de otra manera sería complicado. Pero al tener tantas etiquetas sobre un fondo normalmente oscuro ensucia y satura la imagen con información no siempre útil.

Por estos motivos, la aplicación emplea por defecto la ortoimágen, ya que se presupone que el operario que utiliza la aplicación conoce la zona y no necesita ninguna información extra (la información sobre los elementos que se irán añadiendo se puede consultar tal y como se explica en los puntos posteriores).

El mapa base no suele ser una capa que se cree uno mismo cada vez que se diseñe un aplicación de *Web Mapping* si no que se vale de servicios creados por compañías externas como Google o Bing, o instituciones locales como el IDENA. En este caso se ha optado por utilizar la ortofoto de máxima actualidad que proporciona el IDENA, que sirve la capa como un servicio WMS integrable en la aplicación en el sistema de referencia EPSG: 25830. Si se desea utilizar por ejemplo la de Google hay que tener claro que solamente trabaja con el sistema de referencia EPSG:900913, lo que dificulta un poco más el código ejecutable, ya que todas las capas añadidas se tendrán que transformar.

2.2.4.4. Capas de elementos afectados

Una vez se haya diseñado la base de la aplicación lo siguiente es añadir todas aquellas capas que se han creado y publicado hasta este momento, que son los que verdaderamente le dan sentido al visor de la aplicación. En el momento de integrar estas capas será imprescindible hacerlo de tal manera que se respete el nivel de emergencia.

Las capas a cargar son aquellas que el servidor de mapas las publica en WFS para poder ejecutar acciones más complejas, tal y como se ha expuesto en el apartado '2.2.3. Publicación de las capas', que si se han seguido las instrucciones los nombres se compondrán de: *municipio_nombreDescriptivo.*

Es indispensable conocer cuál es la versión del servicio para poder rellenar adecuadamente las propiedades de la función que se encarga de definir el protocolo de la llamada. Es decir, dependiendo de la versión en el que se haya publicado el WFS, las opciones a rellenar son distintas. Sin embargo el nombre de la capa y la dirección URL son dos de las propiedades básicas y necesarias para el protocolo de llamada (también lo pueden ser el prefijo, que es la dirección URI del espacio de trabajo, y el nombre del campo de la tabla donde se encuentra la geometría para la versión 1.1.0 de WFS). Para que sea posible visualizar solamente los elementos que pertenecen a un nivel de emergencia en concreto se deben aplicar filtros. Estos filtros dependerán del periodo de retorno de los elementos.

Cuando el subsistema de control de NOE indica que el caudal que proporciona el SAIH pone en marcha un cierto Nivel de Alerta la aplicación ofrece los datos correspondientes a ese nivel, que en el caso de las capas están relacionadas con las manchas de inundación: a cada nivel de alerta le corresponde una mancha de inundación concreta.

La tecnología necesaria para esta acción es la de AJAX (*Asynchronous JacaScript And XML*), que se trata de una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones (Baena Carrillo & Ferré, 2011).

Cada cierto tiempo la aplicación realiza una petición de tipo 'GET' al servidor para consultar la base de datos correspondiente cuál es el Nivel de Alerta en la que se encuentra la localidad (mediante JSP en este caso), y cuando este cambie de valor se añaden en la aplicación las actuaciones y capas asociadas a ese nivel concreto.

Como la capa publicada contiene un campo para distinguir a que periodo de retorno corresponde al cada elemento, el filtro se centrará en este campo. Conviene tener cuidado con la comparación 'EQUAL TO' (==) ya que no siempre un valor coincide exactamente con otro aunque a simple vista sí lo parezca. La solución pasa por coger los valores que quedan por encima y por debajo y hacer las comparaciones 'GREATER THAN'

(<) y 'LESS THAN' (>), que aunque puede parecer que complica el código, en muchos de los casos es indispensable hacerlo para que el filtro realmente funcione.

2.2.4.5. Estilos

Cada una de las capas añadidas tiene un estilo por defecto (simbología, forma, color…). Este estilo se irá cambiando dependiendo del elemento y los eventos que se le asigne (añadir pop-ups o relacionarla con objetos fuera del visor), pero en un principio las capas se representan con un determinado estilo que será el de por defecto.

Lo adecuado es asignar un estilo distinto para cada elemento que ayude a identificar el elemento mediante asociación de ideas, es decir, que por ejemplo las manchas de inundación sean de color azul, las parcelas urbanas de color rojo o las calles y carreteras amarillas, siguiendo la simbología establecida para la cartografía por el Instituto Geográfico Nacional, IGN (Subdirección General de Cartografía, 2014).

En cuanto a los puntos de interés, la simbología debe cambiar dependiendo del estado de la actuación a la que este asociada: sin activar, en progreso o finalizado. Además, si se representan dependiendo del tipo, el visor será más intuitivo y resultará más fácil identificar los elementos solamente con un vistazo. Estos estilos por defecto se cambiarán cuando el elemento sea seleccionado y volverán al estilo del principio al deseleccionar para que visualmente sea más fácil comprobar qué elemento ha sido seleccionado.

Una opción (tabla 4) es descargarse iconos de distintos colores desde páginas web especializadas como pueden ser 'MapIcons', 'FlatIcon' o 'IconFinder' para luego poder guardarlas en el servidor de tal manera que se pueda acceder a ellas desde la aplicación dependiendo de lo que necesite. Los estilos iniciales con los que se pintarán estos puntos de interés dependerán del estado en la que se encuentra la actuación en la base de datos que gestiona las emergencias. Para ello se realiza una petición 'GET' de AJAX para conocer el estado y según el valor se le asigna un estilo u otro. Estos etilos iniciales se irán cambiando al mismo tiempo que se realice algún tipo de cambio en los estados (ver apartado 2.2.4.7).

Tabla 4. Ejemplo de los iconos que se pueden emplear para representar puntos de interés (MapIcons)

TIPO	Sin activar	En progreso	Finalizado
Actuación			
Depuradora			
Industria agroalimentaria			
Consistorio			
Buzón			
Centro Comercial			
Instalación Deportiva			
Centro de Educación			

Alojamiento Turístico			
Polígono Industrial			
Centro de Salud Primaria			
Recurso Turístico			
…			

2.2.4.6. Ventanas emergentes para la consulta de información (*pop-ups*)

Una manera de hacer que el mapa sea más comprensible es utilizar ventanas emergentes o *pop-ups* que muestren la información de los elementos (figura 27). La información que se muestra será aquella que se ha decidido mantener en cada capa como campo.

Figura 27. Ejemplo de ventanas emergentes sobre dos elementos: una calle y un punto de interés

Para llevarlo a cabo se le asigna un evento a la capa para que al seleccionarlo (hacer click con el botón izquierdo del ratón) cambie de estilo y salte la ventana emergente con el texto oportuno y quitar el estilo y borrar el *pop-up* al deseleccionarlo (hacer click en cualquier otro sitio del mapa).

2.2.4.7. Interacción entre el visor y el formulario

Como el documento HTML se divide entre el visor y el formulario, resultaría interesante que elementos de uno y otro que estén relacionados interactúen. Es decir, si en el formulario se introducen las actuaciones que se van a llevar a cabo, algunas de ellas estarán representadas mediante elementos puntuales en el propio mapa, y por lo tanto la relación entre ambos (actuación/punto) se podrá materializar empleando cambios de estilos.

En este caso una posibilidad es que al seleccionar un punto en el mapa, si este está relacionado con una actuación (es decir, que contenga un identificador en el campo 'actuacion'), la actuación que le corresponde en el formulario cambie de estilo: cambie levemente de color de fondo, el color de las letras o reescribirlo en mayúsculas. Cualquier cosa que visualmente ayude a identificar dónde se encuentra la actuación. Al deseleccionar el punto lo mejor es que todo vuelva al mismo estilo del principio.

La misma acción se puede llevar a cabo pero en la dirección opuesta, cambiando el color del icono del punto cuando se haga un cambio en el formulario. Estos cambios están relacionados con el estado de la actuación que se ha comentado anteriormente: sin activar, en progreso y finalizado. Para ello el formulario debe permitir cambiar el estado de algún modo y dependiendo del estado en el que se encuentra la actuación cambiar el icono como pueden ser los ejemplos de la tabla 4.

Para que sea posible cambiar el estado de la actuación desde el formulario se deberá incluir algún botón (*radio button* o *submit button* o cualquier otro) que cambie el estado en la base de datos encargada de gestionar las actuaciones y dependiendo de lo que se ha almacenado en cuanto a la actuación cambiar el icono que representa el punto (figura 28). El procedimiento se lleva a cabo en dos pasos: primero se actualiza con una petición 'PUT' (de AJAX) la base de datos que gestiona las emergencias y seguidamente se cambia el estilo del punto en el visor.

Figura 28. Ejemplo de una actuación que mediante un *radio button* cambia el color del icono

2.2.4.8. Comunicación entre el usuario y el desarrollador

Tal y como se ha comentado anteriormente, la aplicación debe permitir al usuario modificar aquellas capas que por algún motivo crea que puedan estar mal: hay que tener en cuenta las diferencias que pueden existir entre el plan teórico y lo que realmente sucede cuando ocurre una inundación.

Para ello se incorporará en la aplicación un control de edición que permitirá crear sobre una capa vacía elementos puntuales, lineales y recintos que podrán ser trasladados o cambiados de forma. Esta capa podrá ser almacenada en la base de datos espacial para que el desarrollador de la aplicación pueda consultarla y que sea él mismo quien modifique la capa que se está publicando. El usuario crea la propuesta de edición y el desarrollador decide cómo modificar las capas originales.

Para almacenar en la base de datos espacial la capa que el usuario ha creado lo más apropiado es hacerlo con técnicas AJAX, mediante la petición 'POST'. Se envía la información (la geometría de la capa editada desde la aplicación) haciendo uso de la operación transaction (un servidor denominado WFS-T) para que el servidor de mapas recoja el XML con los *inserts* que tiene que realizar en la base de datos espacial de la capa vacía que previamente. Esta acción se podrá activar a través de un botón en el formulario. En ese momento tienen que almacenarse automáticamente las nuevas geometrías en la base de datos espacial, y será posible acceder a ella desde el software GIS de escritorio para utilizarlo como referencia.

De este modo el usuario no tiene la posibilidad de cambiar la capas que se visualizan en la aplicación web pero sí que tiene la posibilidad de hacer saber al desarrollador qué zonas son las que realmente se inundan basándose en la experiencia que haya vivido. Al final la última decisión la va a tomar el desarrollador de la aplicación, que podrá cambiar la geometría de la mancha de inundación desde el software GIS de escritorio: se conecta a la base de datos espacial y edita la capa según su criterio.

Una vez se haya modificado la mancha, los elementos afectados (parcelas, calles, puntos de interés…) van a cambiar, por lo que si se está trabajando con tablas y no con vistas, se tendrá que volver a calcular las intersecciones (apartado '2.2.2. Geoprocesamiento'). Para ello se ejecutan de nuevo las consultas SQL eliminando primero las ya existentes para que no se repitan los nombres y así evitar errores.

A partir de ahí, todas las demás herramientas no habrá que modificarlas, ya que al estar todas comunicadas entre sí la capa de la base de datos se visualiza automáticamente en el visor de la aplicación web.

3. RESULTADOS Y DISCUSIÓN

El resultado de aplicar esta metodología de trabajo es un conjunto de procedimientos para cada herramienta GIS que participa en el *Web Mapping*. Es decir, si las herramientas utilizadas son el QGIS, PostGIS, GeoServer y OpenLayers, se han diseñado en los anexos del trabajo procedimientos y códigos ejecutables para que la carga de datos geolocalizados sea lo más rápida y dinámica posible, con el fin de automatizar el proceso en todo lo posible y que así no tenga que depender del municipio en el que se trabaje (figura 29). En el caso de utilizar distintas herramientas GIS para desarrollar una aplicación web de características similares el procedimiento podrá ser igual, solamente quedaría adecuar el código ejecutable y los procedimientos a las nuevas herramientas, los pasos definidos en los anexos serán distintos pero el resultado debería ser parecida a esta.

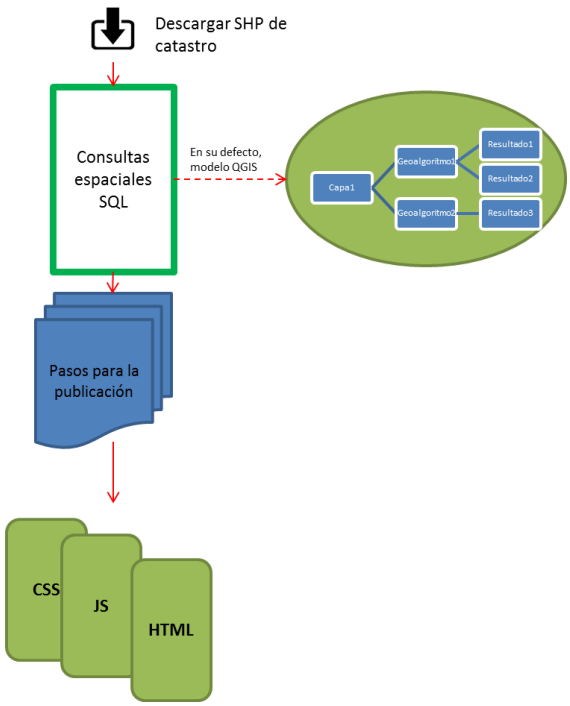


Figura 29. Resultado del trabajo: códigos ejecutables y procedimientos descritos

En cuanto a la metodología descrita, cada procedimiento (pre-procesamiento, geoprocesamiento, publicación y visualización) tendrá sus ventajas y sus inconvenientes a la hora de aplicarlas, pero es importante conocer cuál es el objetivo de cada uno.

La parte inicial, referente al pre-procesamiento de los datos, será la más costosa de todas en lo que a tiempo se refiere. Las capas públicas que puedan ser empleadas en estos trabajos no siempre cumplen con los requisitos deseados, siempre ocurrirán problemas de distinta índole:

- que para que un determinado elemento coincida con la mancha se tenga que modificar su geometría
- que falte algún elemento que se desea analizar pero que no se encuentra disponible en la red porque no existe o porque no es de acceso público
- que como es el caso de las manchas de inundación, los datos sean distintas dependiendo del municipio (habría que intentar mejorar esta cuestión para que el análisis no sea distinto para cada municipio)

Esto conlleva aumentar considerablemente el trabajo manual (que al fin ya al cabo es lo que mayor tiempo requiere) ya que en cada municipio en el que se vaya a implantar la aplicación web necesita una supervisión para ajustar los datos originales a lo que se espera de la aplicación.

Pero por su parte, el hecho de tener que descargarse solamente el catastro de un municipio cada vez que se quiera implantar la aplicación, hace que los pasos siguientes sean más sencillos al no tener que buscar uno a uno los elementos principales de ese municipio.

Una vez se hayan realizado las modificaciones oportunas con la geometría y se tengan descargados los Shapefile necesarios, los siguientes pasos del pre-procesamiento se pueden ejecutar junto a los del geoprocesamiento. El tener un único fichero de consultas SQL, tanto espaciales como las no espaciales, facilita el trabajo y en un único paso se realizan ambos procedimientos. Además el poder realizar las mismas acciones desde un software GIS de escritorio (ejecutando por ejemplo el modelo de procesos creado para QGIS o aplicando los geoalgoritmos uno a uno) permite llevar un cierto control de lo que se va creando y disponer de una alternativa para aquellos casos en el que uno de los dos dé problemas.

Para finalizar, es importante controlar qué campos se dejan y cuáles se eliminan, ya que la visualización final depende de los campos con los que se componen las capas. Por ejemplo, si no se le añade un campo a la capa de los puntos donde se introduzca el identificador de la actuación no será posible relacionar el visor con el formulario, o si no se le añade un campo a cualquiera de las capas que permita relacionarla con una mancha de inundación concreta su visualización no tendrá sentido. Si se desea seguir la manera de trabajar y el código de la aplicación web (HTML, JavaScript y CSS) es preferible que los nombres de los campos coincidan.

La publicación de las capas almacenadas en la base de datos es seguramente uno de los procedimientos más simples de este desarrollo. Lo único que se necesita es crear un nuevo almacén de datos vinculado a un esquema de la base de datos (es decir, uno para cada municipio) y publicar una a una las capas que se desean. En este caso no existe ningún código que automatice el proceso, pero los pasos a seguir son simples y repetitivos.

Por último está la visualización de las capas en una aplicación web. Las posibilidades existentes son tantas que lo que se ha descrito en este informe solamente puede considerarse un ejemplo para trabajar, ya que las funcionalidades que se le pueden llegar a añadir son muchas más. Lo mismo ocurre con los estilos: los colores, tamaños, iconos… puestos como ejemplo en los anexos pueden ser cambiados para ajustarlos a las necesidades de la aplicación.

Cada desarrollador puede y tiene que diseñar la aplicación web a su estilo. En general todo el código se puede considerar modificable si es para ajustar la aplicación a las necesidades del usuario. Si se decide seguir el código descrito en un nuevo municipio, solamente ha de cambiarse el nombre de las capas (*municipio_nombreDeLaCapa*) y las coordenadas del centro y límites de la extensión del mapa.

La integración de este visor en la aplicación web NOE todavía queda por completar. Aunque sí que es cierto que el código creado está pensado exclusivamente para que su integración sea lo más sencilla posible, aún quedan por mejorar ciertos aspectos sobre la relación entre el visor y el gestor de emergencias.

Esta integración completa en la aplicación web NOE se conseguiría con la tecnología AJAX, que permite mantener una comunicación asíncrona en segundo plano con el servidor. En este caso se utilizaría para cargar los datos dependiendo del Nivel de Alerta establecido y determinar los estilos de los puntos consultando el estado en la base de datos para la gestión de emergencias. Las URL de las peticiones quedan a cargo de los desarrolladores de la aplicación que en estos momentos están en la empresa.

Otra funcionalidad que queda por finalizar es la que se encarga de guardar la capa creada por el usuario en la base de datos espacial para que pueda ser consultada por el desarrollador. Con esta función, junto a otras, una aplicación web puede llegar a ser un GIS completo, ya que directamente desde el navegador se podrían crear, editar y guardar capas con información geográfica.

Para terminar con el visor, comentar que al trabajar con distintas capas solamente se puede consultar la que se encuentra más arriba, es decir, que si se quiere sacar la ventana emergente, por ejemplo, de una calle todas las capas que se encuentran encima se tendrán que desactivar. Pero solucionar este problema no es tarea fácil: por una parte se pueden meter todos los elementos en una única capa pero eso haría que el visor no fuera tan usable, otra opción es crear un control en el formulario que permita poner arriba del

todo una capa concreta pero el formulario se llenaría de controles convirtiéndolo en menos amigable y por último se podrían recorrer todas las capas cuando se haga click hasta encontrar un elemento en alguna de ellas, esta solución requeriría conocimientos altos de programación.

Cuando todo el sistema de carga de datos está activado, desde la propia descarga de datos ya existentes, como pueden ser el catastro o las manchas de inundación para cada periodo de retorno, hasta su integración en la aplicación web NOE los datos pasan por distintas herramientas GIS que están en constante comunicación entre ellos, permitiendo visualizar en un entorno web capas que se pueden modificar directamente en el software GIS de escritorio.

Aunque esta fluidez en el trabajo dependerá mucho del servidor y la arquitectura montada en ella. Si el servidor no cumple con los requisitos apropiados las herramientas conectadas entre sí no trabajarán con la suficiente fluidez y el trabajo se convierte más lento y costoso. La base de datos espacial, que al fin y al cabo es el eje de la arquitectura, debe garantizar que estará disponible en todo momento y con una velocidad suficiente. Lo mismo ocurre con el servidor de mapas.

Otro de los aspectos a tener en cuenta en esto de la fluidez y la integridad del sistema, es el uso de vistas en la base de datos. Se ha mencionado que se consigue mayor integridad con las vistas (un cambio en la tabla original repercute directamente en la vista) pero la ejecución se hace lenta y difícil de gestionar, mientras que con las tablas ocurre todo lo contrario: no existe integridad pero son más fáciles de gestionar.

Con el fin de superar este dilema se podrían analizar las ventajas e inconvenientes de usar vistas materializadas, que son una especia de híbridos entre ambas. Su aplicación se realizaría cambiando el código de la consulta SQL de *CREATE TABLE* o *CREATE VIEW* a *CREATE MATERIALIZED VIEW*.

De momento, con el sistema planteado en estos momentos con tablas, la falta de integridad tampoco supone un problema mayor. Las tablas solamente van a ser actualizadas cuando ocurra una inundación y el usuario haya decidido que la mancha de actuación se puede mejorar, guardando para ello una capa creada en la aplicación en la base de datos para que el desarrollador modifique la capa original. Se trataría de actualizar desde el software GIS de escritorio la capa de las manchas original y cambiar manualmente las capas derivadas (calles afectadas, parcelas afectadas…) en la base de datos espacial, un procedimiento que lleva 10 minutos cuando ocurra una inundación. Aun así sí que es cierto que tener un sistema integrado facilita el trabajo.

No hay que olvidar que en realidad esta aplicación sirve para plasmar un Plan de Actuaciones que en la mayoría de los casos no existe y por lo tanto se tienen que redactar y conseguir el visto de bueno del Departamento de Protección Civil de Navarra. El poder determinar los elementos geográficos afectados en la primera parte del desarrollo (pre-procesamiento + geoprocesamiento) puede servir de gran ayuda para esa redacción del Plan ampliando las funcionalidades de estos procedimientos y el objetivo principal del trabajo.

Otra de las mejoras a futuro es la versión de OpenLayers a emplear. Todo el código descrito hace referencia a la versión 2, pero esta versión ya se está dejando a un lado poco a poco, y aunque todavía sigue siendo la librería más utilizada, su desarrollo y mejora se ha dejado para sacar adelante la versión 3. En un futuro la aplicación podría ser mejorada migrando el código de la versión antigua a esta nueva versión.

El sistema de carga de datos queda totalmente abierto a incluir más capas distintas a las mencionadas en el trabajo. Si se considera que podría resultar interesante disponer de algún otro elemento geográfico su inclusión en la aplicación puede ser posible, pudiendo tomar como ejemplo los procedimientos seguidos en este trabajo, sobre todo para determinar qué campos dejar o eliminar y conocer cómo visualizarlos en la aplicación web.

4. CONCLUSIONES

El objetivo establecido al principio del trabajo de diseñar un sistema de carga de datos geolocalizados se ha cumplido en su mayoría. La arquitectura web GIS necesaria para el desarrollo de una aplicación de *Web Mapping* ha sido definida y los códigos ejecutables y procedimientos descritos a lo largo del informe detallan los pasos a seguir en cada una de las herramientas GIS instaladas. El visor resultante del diseño (figura 30), que es en realidad la parte donde se materializa de una forma visual el proceso que se ha seguido desde la descarga de los datos hasta su implementación en una aplicación web, consta de las suficientes capas y funcionalidades como para que la aplicación sea lo más clara e intuitiva posible con una interacción constante con el formulario al que acompaña.

Figura 30. Ejemplo de la aplicación desarrollada (sin integrar en la aplicación web NOE)

Sin embargo, la integración de la visualización de los datos en la aplicación web NOE de Tesicnor queda pendiente del desarrollo de ciertas partes del código referentes a la interactuación con la base de datos que gestiona las emergencias, que es la parte de la aplicación que no está relacionada con el GIS y lo desarrollan los programadores de Tesicnor.

Este diseño es un prototipo que se ha comprobado que funciona y que sirve para cumplir con el objetivo, pero esto no significa que no se pueda modificar, ampliar o eliminar los procedimientos o funcionalidades expuestas. Tener este desarrollo como ejemplo permite ajustar en un futuro el sistema de carga de datos a las necesidades que la aplicación necesite. No se trata de que este sea único y estático, si no que sea posible introducir más capas, funcionalidades, campos de capas (para que aparezcan en las ventanas emergentes)… Entendiendo lo que se hace en cada una de las etapas es posible cambiar el resultado.

En este tipo de trabajos, aunque se intente automatizar en todo lo posible el sistema de la carga de datos, el trabajo manual y la supervisión por parte de un operario resulta indispensable, sobre todo en algunas etapas del proceso, como es el caso del pre-procesamiento. Es probable que algunos de los conjuntos de datos que se consulten no se ajusten del todo a las necesidades del momento, lo que hace imprescindible modificar o crear las geometrías a mano.

Por otra parte, el trabajar con herramientas libres de código abierto instaladas en un servidor propio sin alojar nada en la nube, permite que el control sea absoluto sobre lo que se hace en cada herramienta GIS utilizada. Este control absoluto tiene la ventaja de que cualquier funcionalidad o proceso puede ser añadida y modificada según se quiera para adaptarse a lo planificado, pero tiene como inconveniente el

tener que hacer frente a grandes cantidades de código y programación, es decir, un conocimiento técnico más amplio. Por esta última razón trabajos como este ayudan a profundizar y mejorar en conocimientos relacionados con la programación y desarrollo de webs.

En este aspecto, la arquitectura web GIS básica queda implementada en la empresa para que en un futuro sea posible desarrollar más aplicaciones web GIS, ya que los GIS en general son herramientas aplicables a distintos campos y disciplinas: cualquier dato que de alguna u otra manera se puede relacionar con una ubicación espacial puede ser gestionada por estas herramientas.

Aun así, al igual que pasa con todo lo relacionado con la informática, aunque se haya implantado la arquitectura web GIS es necesario estar al tanto de las últimas novedades y actualizaciones de las herramientas que rodean el mundo del *Web Mapping*. En este sentido, tal y como se comentado antes, la versión 3 de OpenLayers se está convirtiendo poco a poco más usada que la versión anterior, centrándose actualmente todos los esfuerzos en la versión nueva. Habría que plantearse migrar en un futuro. Otro ejemplo es el de CartoDB, que desde que se empezó el trabajo hasta que se ha finalizado ha cambiado el nombre a Carto. Habría que mirar si también han cambiado las funcionalidades y condiciones de uso.

Si se decide que se va a continuar con los análisis GIS o los visores integrables en aplicaciones, y si se decide que las herramientas instaladas satisfacen las necesidades, se podría plantear contratar un servicio de soporte y mantenimiento que ofrecen plataformas geoespaciales completas como el OpenGeo Suite.

La potencialidad de estas aplicaciones web GIS también está relacionada con las distintas tecnologías y técnicas que existen en torno al desarrollo web. Si a las capacidades que ofrecen las API que trabajan con datos espaciales se les añade tecnologías como el AJAX, PHP, .NET, etc. los visores creados estarán insertados en aplicaciones web con capacidad de trabajar más rápido, en contacto con la base de datos que a su vez se conectan en tiempo real con sensores…

Al final llega un momento en el que la aplicación creada cumple las funciones de un GIS. A través de estas aplicaciones web es posible crear, editar, gestionar, analizar, procesar, borrar y guardar en bases de datos espaciales cualquier tipo de dato geolocalizado desde un simple navegador web sin tener que tener instalado ningún software en el disco duro local. Con la conexión entre el servidor y el navegador web es posible crear una especie de GIS (más sencillo o más completo) que el usuario pueda manejar sin necesidad de un conocimiento técnico muy alto. En este sentido, el avance en las Infraestructuras de Datos Espaciales (IDE) y los servicios OGC han contribuido enormemente en está popularización de la información geográfica, permitiendo por una parte trabajar con datos creados por distintas fuentes y por otro facilitando que las capas almacenadas en bases de datos puedan ser empleadas en aplicaciones web.

En el caso de la aplicación diseñada en este trabajo el usuario no es que tenga la posibilidad de realizar complejos estudios espaciales, pero sí que tiene la oportunidad de enviar propuestas en forma de capa vectorial. La posibilidad que proporcionan los WFS-T (o cualquier otro sistema de acceder a la base de datos espacial) de guardar, editar, eliminar o modificar las capas de la base de datos espacial u otros servicios como el W*eb Processing Service* (WPS) son una de las claves para que se empiece a cuestionar la necesidad de los softwares GIS de escritorio en un futuro.

Además, hoy en día las bases de datos espaciales ya realizan geoprocesamientos complejos y de manera rápida, dejando casi los softwares GIS de escritorio en un segundo plano. Sin embargo, en este trabajo sí que se han utilizado y recomendado los softwares GIS de escritorio para el desarrollador en el sistema de carga de datos, sobre todo para visualizar y comprobar las operaciones realizadas por la base de datos y sobre todo para modificar las geometrías de manera más fácil.

Como última conclusión resaltar la necesidad de tocar en mayor o menor medida todas las disciplinas dentro de los GIS (GIS de escritorio, bases de datos espaciales, IDEs y *Web Mapping*) para conectarlos entre sí con el fin de conseguir una aplicación web GIS completo. Cada una de las especialidades relacionadas con GIS impartidas en el Máster Universitario en Sistemas de Información Geográfica y Teledetección ha servido para completar alguna de las etapas del proyecto.

5. BIBLIOGRAFÍA

Alouene, Y., & Petropoulos, G. P. (2013). Flooded area cartography and damage assessment from the combined use of Landsat TM and ANNs. *Computers & Geosciences*, *15*(April 2013), 14069.

Auynirundronkool, K., Chen, N., Peng, C., Yang, C., Gong, J., & Silapathong, C. (2012). Flood detection and mapping of the Thailand Central plain using RADARSAT and MODIS under a sensor web environment. *International Journal of Applied Earth Observation and Geoinformation*, *14*(1), 245–255. http://doi.org/10.1016/j.jag.2011.09.017

Baena Carrillo, M., & Ferré, M. (2011). *Creación de una aplicación SIG con OpenLayers , ExtJS y MySQL*. Universitat Rovira I Virgili.

Ballatore, A., McArdle, G., Tahir, A., & Bertolotto, M. (2011). A Comparison of Open Source Geospatial Technologies for Web Mapping. *International Journal of Web Engineering and Technology*, *6*(4), 354–374.

Boletín oficial del Estado, B. Real Decreto 903/2010, de 9 de julio, de evaluación y gestión de riesgos de inundación (2010). España.

Boundless. (2016). Boundless Official (OpenGeo Suite). Retrieved June 18, 2016, from http://boundlessgeo.com/products/opengeo-suite/

CartoDB. (2016). CartoDB Documentation. Retrieved May 15, 2016, from https://cartodb.com/docs/

Daras, G., Agard, B., Cambazard, H., & Penz, B. (2015). Development of business spatial analysis tools: Methodology and framework. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *48*(3), 1894–1899. http://doi.org/10.1016/j.ifacol.2015.06.363

Dottori, F., Salamon, P., Bianchi, A., Alfieri, L., Hirpa, F., Feyen, L., & Hirpa, F. A. (2016). Development and evaluation of a framework for global flood hazard mapping. *Advances in Water Resources*, *94*(16), 87–102. http://doi.org/10.1016/j.advwatres.2016.05.002

Elkhrachy, I. (2015). Flash Flood Hazard Mapping Using Satellite Images and GIS Tools: A case study of Najran City, Kingdom of Saudi Arabia (KSA). *The Egyptian Journal of Remote Sensing and Space Science*, *18*(2), 261–278. http://doi.org/10.1016/j.ejrs.2015.06.007

EU. (2007). DIRECTIVE 2007/60/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. *Official Journal of the EU*, *31*(November 2007), 8.

Evangelidis, K., Ntouros, K., Makridis, S., & Papatheodorou, C. (2014). Geospatial services in the Cloud. *Computers and Geosciences*, *63*, 116–122. http://doi.org/10.1016/j.cageo.2013.10.007

Fernandes, A., Goulão, M., & Rodrigues, A. (2013). A Comparison of Maps Application Programming Interfaces. *16th AGILE Conference on Geographic Information Science*. Retrieved from http://arxiv.org/abs/1305.3485

Gastesi, R. (2016). *Plan Municipal de Emergencias ante inundaciones. Ejemplo en el Valle de Baztan*. *TRACASA*. Retrieved from http://www.tesicnor.com/jornada-demostrativa-solucion-noe/

GeoExt. (2016). GeoExt Documentation. Retrieved May 15, 2016, from http://geoext.org/v1/docs.html

Google. (2016). Javascript Google Maps API Documentation. Retrieved May 15, 2016, from https://developers.google.com/maps/documentation/javascript/?hl=es

Haklay, M. (2010). How good is volunteered geographical information? A comparative study of OpenStreetMap and ordnance survey datasets. *Environment and Planning B: Planning and Design*, *37*(4), 682–703. http://doi.org/10.1068/b35097

INSPIRE. (2016). INSPIRE GEOPORTAL: enhancing access to European spatial data. Retrieved June 25, 2016, from http://inspire-geoportal.ec.europa.eu/

Ip, F., Dohm, J. M., Baker, V. R., Doggett, T., Davies, A. G., Castaño, R., … Rabideau, G. (2006). Flood detection and monitoring with the Autonomous Sciencecraft Experiment onboard EO-1. *Remote Sensing of Environment*, *101*(4), 463–481. http://doi.org/10.1016/j.rse.2005.12.018

Leaftlet. (2016). Leaflet Documentation. Retrieved May 15, 2016, from http://leafletjs.com/reference.html

MAGRAMA. (2016). Gestión de los riesgos de inundación. Retrieved May 15, 2016, from http://www.magrama.gob.es/es/agua/temas/gestion-de-los-riesgos-de-inundacion/

MapBox. (2016). MapBox Official. Retrieved May 15, 2016, from https://www.mapbox.com/

Mason, D. C., Giustarini, L., Garcia-Pintado, J., & Cloke, H. L. (2014). Detection of flooded urban areas in high resolution Synthetic Aperture Radar images using double scattering. *International Journal of Applied Earth Observation and Geoinformation*, *28*(1), 150–159. http://doi.org/10.1016/j.jag.2013.12.002

Ministerio de Fomento. (2016). Infraestructura de Datos Espaciales de España: el portal de acceso a la información geográfica de España. Retrieved June 25, 2016, from http://idee.es/

Ogilvie, A., Belaud, G., Delenne, C., Bailly, J. S., Bader, J. C., Oleksiak, A., … Martin, D. (2015). Decadal monitoring of the Niger Inner Delta flood dynamics using MODIS optical data. *Journal of Hydrology*, *523*, 368–383. http://doi.org/10.1016/j.jhydrol.2015.01.036

Olaya, V. (2013). *Sistemas de Información Geográfica*. *Journal of Chemical Information and Modeling* (Vol. 53). http://doi.org/10.1017/CBO9781107415324.004

OpenLayers. (2016). OpenLayers Documentation. Retrieved May 15, 2016, from http://openlayers.org/en/latest/doc/

PostGIS. (2016). PostGIS Documentation. Retrieved July 5, 2016, from http://postgis.net/docs/

Pusineri, G. (2004). Aplicación de Sistemas de Información Geográfica para la Prevención de Riesgos y la Formulación de Planes de Contingencia en Inundaciones, 1–13. Retrieved from http://www.crid.or.cr/digitalizacion/pdf/spa/doc15740/doc15740.htm

SITNA. (2016a). Geoportal de Navarra. Retrieved June 10, 2016, from http://sitna.navarra.es/geoportal/

SITNA. (2016b). Manual de usuario del Visor geográfico de la API SITNA. Retrieved from http://sitna.navarra.es/geoportal/recursos/Manual usuario Visor API SITNA.pdf

Smith, D. A. (2016). Online interactive thematic mapping: Applications and techniques for socio-economic research. *Computers, Environment and Urban Systems*, *57*, 106–117. http://doi.org/10.1016/j.compenvurbsys.2016.01.002

Sohn, H., Heo, J., Yoo, H., Kim, S., & Cho, H. (2008). Hierarchical multi-sensor approach for the assessment of flood related damages. *Proceedings of the XXI Congress of …*, 207–210. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.184.1659&rep=rep1&type=pdf

Song, X. feng, Rui, X. ping, Hou, W., & Tan, H. qiao. (2008). An OGC standard-oriented architecture for distributed coal mine map services. *Journal of China University of Mining and Technology*, *18*(3), 381–385. http://doi.org/10.1016/S1006-1266(08)60080-2

Steiniger, S., & Hay, G. J. (2009). Free and open source geographic information tools for landscape ecology. *Ecological Informatics*, *4*(4), 183–195. http://doi.org/10.1016/j.ecoinf.2009.07.004

Subdirección General de Cartografía, (IGN). (2014). *Normas de edición del MTN25* (Versión 1.3).

Swain, N. R., Latu, K., Christensen, S. D., Jones, N. L., Nelson, E. J., Ames, D. P., & Williams, G. P. (2015). A review of open source software solutions for developing water resources web applications. *Environmental Modelling & Software*, *67*, 108–117. http://doi.org/10.1016/j.envsoft.2015.01.014

Tesicnor. (2016a). Gestor de planes de emergencia para inundaciones: NOE. Retrieved May 15, 2016, from http://www.noe.tesicnor.com/

Tesicnor. (2016b). Página Oficial de Tesicnor. Retrieved May 15, 2016, from http://www.tesicnor.com/

ANEXOS

ANEXO I. CREAR UN MAPA SIMPLE DE PRUEBA CON CADA UNA DE LAS HERRAMIENTAS ANALIZADAS

Uno de las maneras para poder comparar las distintas herramientas analizadas (Google Maps API, Openlayers, Leaflet, CartoDB y MapBox) es crear un mapa en la web con cada una de ellas. Al ser a modo de prueba, basta con un mapa simple pero representativo. En este caso se ha decido incorporar un mapa base junto a una capa vectorial descargada desde una IDE (Anexo II) que se encuentre en local y otra capa proveniente de un Web Map Service (WMS), en este caso de la IDE de Navarra (IDENA).

Para poder trabajar con estas herramientas siempre se utilizan las APIs que proporcionan, que pueden descargarse directamente o acceder a ellas mediante una conexión de internet. Las URL de las páginas web para las descargas son las siguientes:

- La API de Google Maps no tiene posibilidad de descarga, por lo que es necesario inscribirse primero en su página web oficial y así disponer de una clave que sirve solo para una única URL. El código de JavaScript para la Google Maps API se carga a través de una URL de *arranque* con la forma https://maps.googleapis.com/maps/api/js. Sin embargo, esta API ya la están utilizando en la empresa, y no hace falta hacer pruebas con ella.

- Leaflet: de su propia página http://leafletjs.com/download.html

- Openlayers: de su propia página http://openlayers.org/download/

- Cartodb.js: de GitHub https://github.com/CartoDB/cartodb.js/

- Mapbox.js: de GitHub https://github.com/mapbox/mapbox.js/

La idea es ir probando con distintas herramientas las distintas maneras y capas que puedan servir para ir completando la cartografía web: librería en local/en línea, mapa base imagen satelital/cartografía base, capa de inundaciones 'Áreas de riesgo'/'Periodos de retorno', capa 'Parcelas urbanas'/'Manzanas'…

OPENLAYERS 2

OpenLayers.js al ser una de las librerías que más cerca de las funciones trabaja consta de un único zip donde se acumulan todas las funcionalidades, y esto conlleva el tener que escribir más código. En este caso se trabajará con la librería descargada desde su página web oficial ubicada en local.

Los pasos a seguir son sencillos en este caso, lo único es crear un mapa y cargar dos capas cada una con una función distinta: una para visualizar una capa en WMS y otra para cargar un archivo vectorial que se encuentre en un directorio conocido. Lo siguiente es añadirlas al mapa y centrar la vista.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Openlayers2</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <!-- Inluir LibreríaOpenLayers v2.11 de la web, o en local-->
        <script type="text/javascript" src="http://openlayers.org/api/2.13/OpenLayers.js"></script>
        <!-- Inluir un estilo general -->
        <link rel="stylesheet" type="text/css" href="./css/General.css">
        <!-- Inluir un estilo para elementos de esta página -->
        <link rel="stylesheet" type="text/css" href="./css/UnMapa.css">
    </head>
    <body onload="init()">
        <div id="Mapa1"></div>
        <script>
        function init() {
            //Crear mapa en el elemento DOM
            var mapa = new OpenLayers.Map("Mapa1");
            //Crear capa raster IDENA
            var base = new OpenLayers.Layer.WMS("Ortofoto", "http://idena.navarra.es/ogc/wms",
                {
                    layers: "IDENA:ortofoto_maxima_actualidad ",
                    transparent: true,
                    format: 'image/png'
```

```
                },
                {
                    isBaseLayer: true
            });
            //Crear capa WMS
            var wms = new OpenLayers.Layer.WMS("IDENA Parcelas Urbanas",
                "http://idena.navarra.es/ogc/wms",
                {
                    layers: "IDENA:CATAST_Pol_ParcelaUrba",
                transparent: true,
                format: 'image/png'
                },
                {
                isBaseLayer: false
                });

            //Crear capa vectorial
            var json = new OpenLayers.Layer.Vector("Retorno",{
                protocol: new OpenLayers.Protocol.HTTP({
                    url: "./res/retorno.kml",
                    format: new OpenLayers.Format.KML({
                        extractStyles: true,
                        extractAttributes: true
                    })
                }),
                strategies: [new OpenLayers.Strategy.Fixed()]
            });
            //Añadir capas al mapa
            mapa.addLayers([base, wms, json]);
            //Hacer zoom hasta el tamaño máximo
            mapa.setCenter(new OpenLayers.LonLat(-1.65,42.82), 15);
        };
        </script>
    </body>
</html>
```

El resultado es el siguiente:

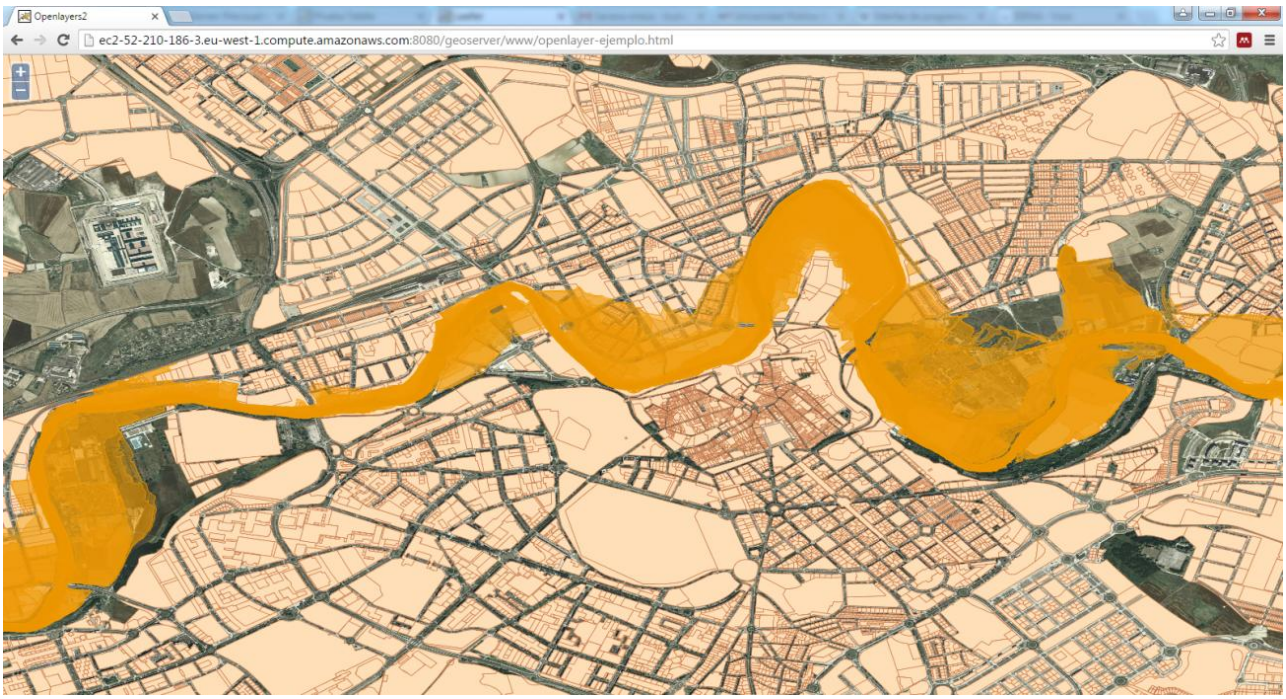


Figura 31. Mapa creado con OpenLayers

LEAFLET

Leaflet es una librería muy ligera con funcionalidades básicas al que se le tiene que añadir plugins para las funcionalidades extras. Es una de las más sencillas y CartoDB como MapBox están construidas sobre ella (al fin y al cabo son sus plugins), por ese motivo parte del código que se muestra aquí también servirá para desarrollar en CartoDB y MapBox.

Como ya se ha dicho, la librería recoge las funcionalidades básicas como inicializar un mapa o añadir una capa base pero en este caso se tendrán que ampliar con alguna librería para poder importar archivos vectoriales (las llamadas a WMS sí que están incorporadas en la librería). El plugin usado, aunque existan otras alternativas, es 'Omnivore'.

Los pasos a seguir son idénticos al de OpenLayers, lo único que cambia es el lenguaje. En este caso las capas se cargan y se añaden en la misma línea, pero por lo demás los pasos vuelven a ser: llamar a todas las librerías que se pretenden utilizar, crear un mapa centrando la vista, añadir una capa base y posteriormente añadir ambas capas, primero el vectorial en formato KML y después el WMS.

```
<!DOCTYPE html>
<html>
    <head>
      <title>Leaflet</title
      <!-- http://maptimeboston.github.io/leaflet-intro/ -->
      <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css"/>
      <script src="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js"></script>
      <script src='//api.tiles.mapbox.com/mapbox.js/plugins/leaflet-omnivore/v0.3.1/leaflet-
omnivore.min.js'></script>
      <script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>

      <style>
        #map{ height: 100% }
      </style>
    </head>
    <body>
      <div id="map"></div>

      <script>

      // Inicializar el mapa
      var map = L.map('map').setView([42.82, -1.65], 15);

      // Añadir capa base
      L.tileLayer('http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',{
                    maxZoom: 20,
                    subdomains:['mt0','mt1','mt2','mt3']
                }).addTo(map);

      // Añadir capa vectorial
      omnivore.kml('./res/retorno.kml').addTo(map);

      //Añadir capa WMS
        var retorno = L.tileLayer.wms('http://idena.navarra.es/ogc/wms', {
            format: 'image/gif',
            transparent: true,
            layers: 'IDENA:CATAST_Pol_ParcelaUrba'
        }).addTo(map);

      </script>
    </body>
</html>
```

El resultado es el siguiente:

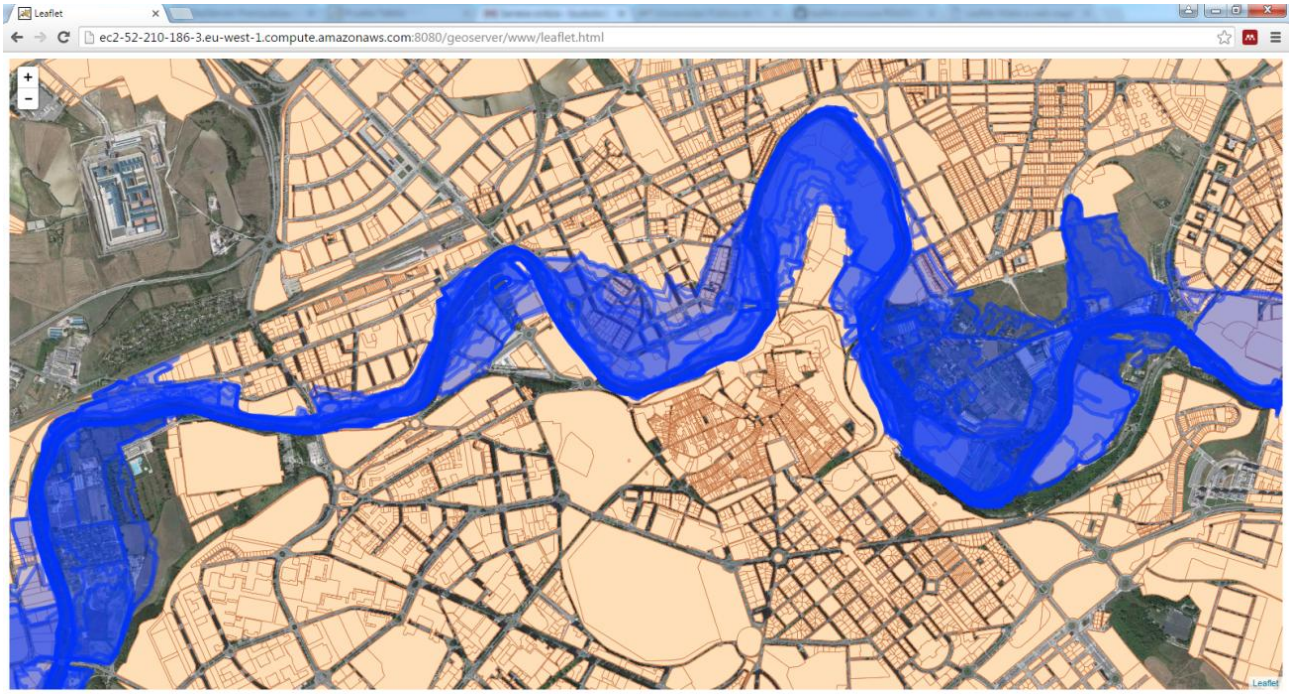


Figura 32. Mapa creado con Leaflet

CARTODB

Una de las ventajas que proporciona el CartoDB es su editor en línea, ya que permite diseñar primero los estilos, capas, elementos… que se visualizarán y una vez se tenga todo listo publicarlo para que sea posible después acceder a ella. Es decir, se diseña todo en su Editor que internamente lo va guardando en la nube para que después, mediante el CartoDB.js, poder llamar a ese mapa ya creado (como si se tratara de un mapa base de Google, por ejemplo) e insertarlo en la página web.

Cuando se trabaja en el Editor, todo lo que se va cargando y creando es insertada en un PostgreSQL con PostGIS y por lo tanto será posible tratar los datos como si fuesen tablas y además realizar consultas SQL sobre las mismas, que es otra de las potencialidades que tiene esta herramienta.

Para empezar a trabajar lo primero es crear una cuenta en CartoDB (en su página web oficial) y así poder acceder al propio Editor. El Editor prácticamente se divide en dos grandes partes: conjuntos de datos y mapas (aunque también existen más apartados como pueden ser el gestor de la cuenta o la 'key' para la API).

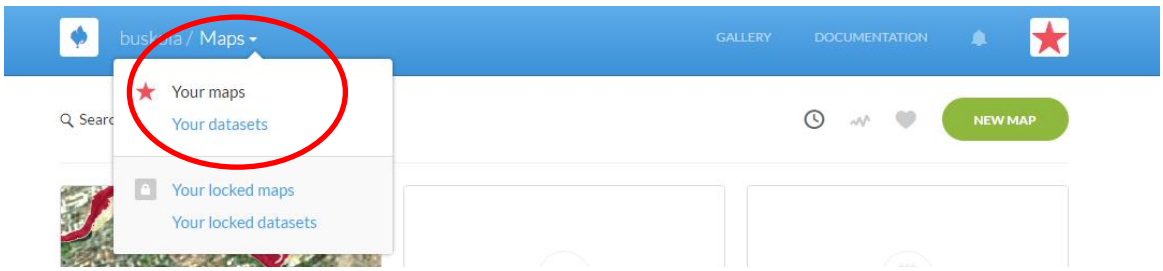


Figura 33. Desplegable para elegir si se trabaja con los conjunto de datos o con mapas en CartoDB Editor

Para poder añadir cualquiera de los dos (datos o mapa) solamente hace falta clicar el botón 'New …' que aparece arriba a la derecha:

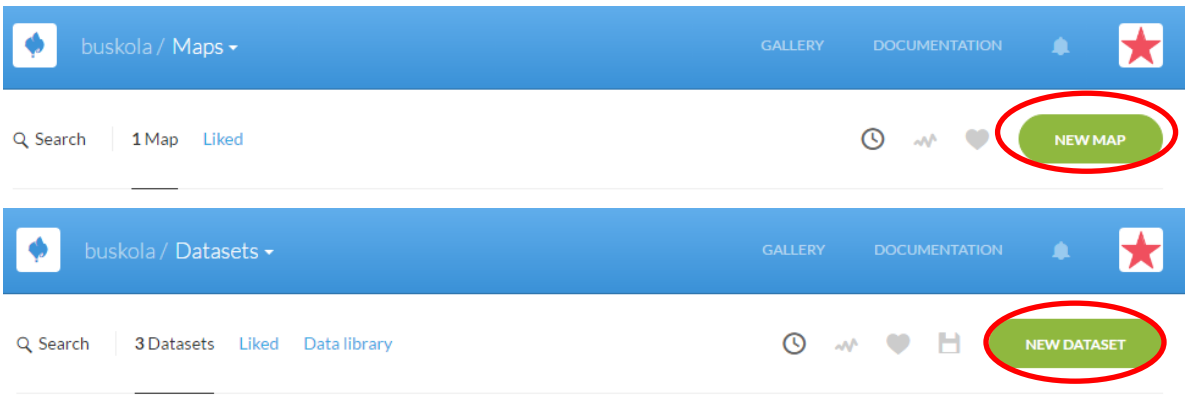


Figura 34. Botones para crear un nuevo mapa y un nuevo conjunto de datos en CartoDB Editor

Primero se cargan los conjuntos de datos que se vayan a utilizar: CSV, XLS, ZIP (Shapefile), KML, GPX; Google Drive; Dropbox; Box; Twitter… para después insertarlos en un nuevo mapa (figura 35).

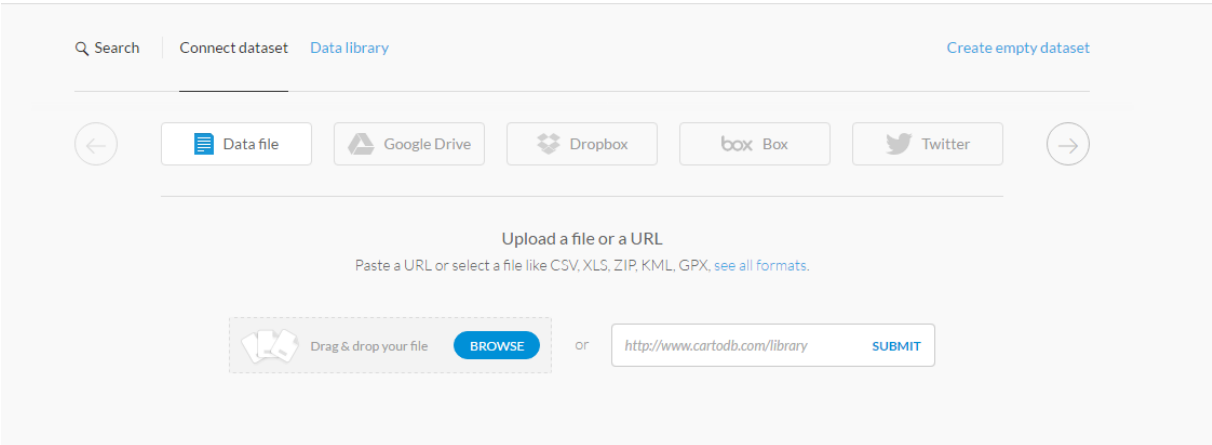


Figura 35. Interfaz para la carga de conjunto de datos en CartoDB Editor

No hay que olvidar que es un servicio de pago y que la prueba gratuita solamente deja cargar 4 capas que ocupen entre ellas 250 MB. Para controlar que no se salta ese límite, es posible hacer el seguimiento en la pestaña del usuario:

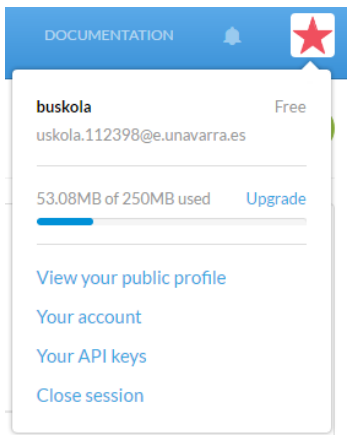


Figura 36. Desplegable del usuario de CartoDB Editor

A continuación se crea un nuevo mapa pulsando el botón señalado anteriormente en la figura 34, que pide qué datasets se quieren añadir en un principio al mapa. Las capas de un mapa se podrán quitar y añadir en cualquier momento, por lo que no pasa nada si se olvida alguna.

Una vez creado el mapa se podrá acceder a lo que sería al Editor en sí, la interfaz gráfica para el manejo de datos (figura 37). Para ello es suficiente con clicar en el mapa que se acaba de crear.

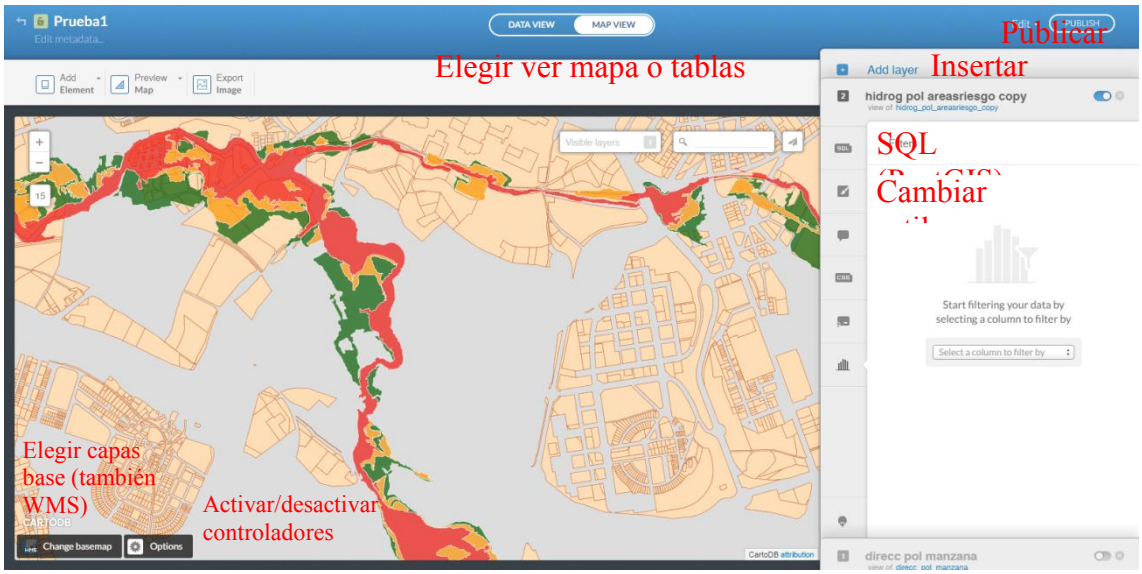


Figura 37. Interfaz del Editor de CartoDB (señaladas las funcionalidades más utilizables)

Las funcionalidades son muchas: se pueden visualizar tanto el mapa que se va creando como las tablas de las capas (donde se pueden filtrar, añadir o eliminar tanto atributos como registros…), añadir mapas base de imágenes aéreas como callejeros y también WMS (es el único modo de añadirlas, por lo que no permite tener de fondo un mapa base y encima una capa WMS), cambiar estilos, añadir controladores (selector de capas, botones zoom, leyendas…) o texto, hacer consultas SQL sobre PostGIS…

Con todas estas posibilidades primero se diseña el mapa en su totalidad (aunque siempre es posible modificarlo desde las distintas librerías proporcionadas por CartoDB) después será posible publicarlo (figura 38): mandar como link, embeber en una página web o llamarlo desde el CartoDB.js.

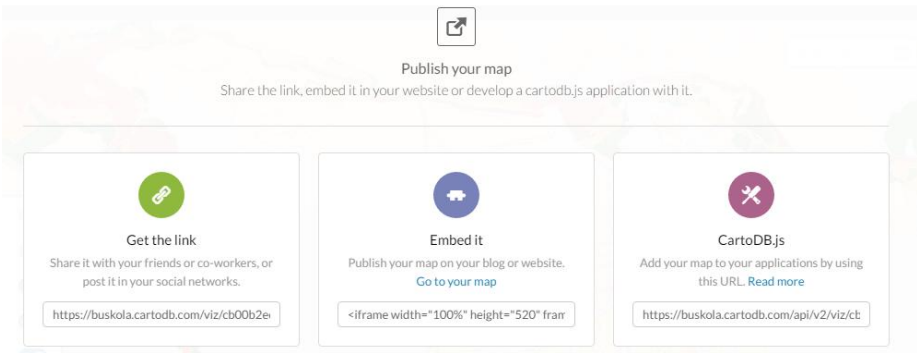


Figura 38. Distintas maneras de publicar el mapa creado en CartoDB

En este caso, lo que interesa es poder utilizar el mapa creado por el CartoDB.js y así comprobar que se puede utilizarlo para insertarlo en una aplicación web.

La estructura del archivo HTML + JavaScript que se utilizaría para que el servidor se lo envíe al navegador web cuando este le haga la petición será la siguiente. Las partes del código más importantes son estas:

- Llamar al CartoDB.js y CartoDB.css (estilo) en la cabecera, en este caso sin haberlos descargado directamente.
- Crear un mapa siguiendo la estructura de Leaflet y añadirle una capa base.
- Añadir lo creado en el Editor copiando la URL en la variable 'vizjson' para utilizarlo en la función *.createLayer*.

```
<!DOCTYPE html>
<html>
    <head>
        <title>CartoDB</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <script src="http://libs.cartocdn.com/cartodb.js/v3/3.15/cartodb.js"></script>
        <link rel="stylesheet"
href="http://libs.cartocdn.com/cartodb.js/v3/3.15/themes/css/cartodb.css" />
        <style>
          html, body, #map {
            height: 100%;
            padding: 0;
            margin: 0;
          }
        </style>
    <body>
        <div id="map"></div>
        <script type="text/javascript">
            window.onload = function() {
                var options = {
                    center: [42.82, -1.65],
                    zoom: 14
                  }
                var map_object = new L.Map('map', options);
                //Añadir mapa base
                L.tileLayer('http://{s}.google.com/vt/lyrs=s&x={x}&y={y}&z={z}',{
                    maxZoom: 20,
                    subdomains:['mt0','mt1','mt2','mt3']
                }).addTo(map_object);
                //Añadir diseño creado en el Editor
                var vizjson = 'https://buskola.cartodb.com/api/v2/viz/cb00b2ee-168f-11e6-ab32-
0e787de82d45/viz.json';
                cartodb.createLayer(map_object, vizjson).addTo(map_object);
            }
        </script>
    </body>
</html>
```

Se obtiene el siguiente resultado:

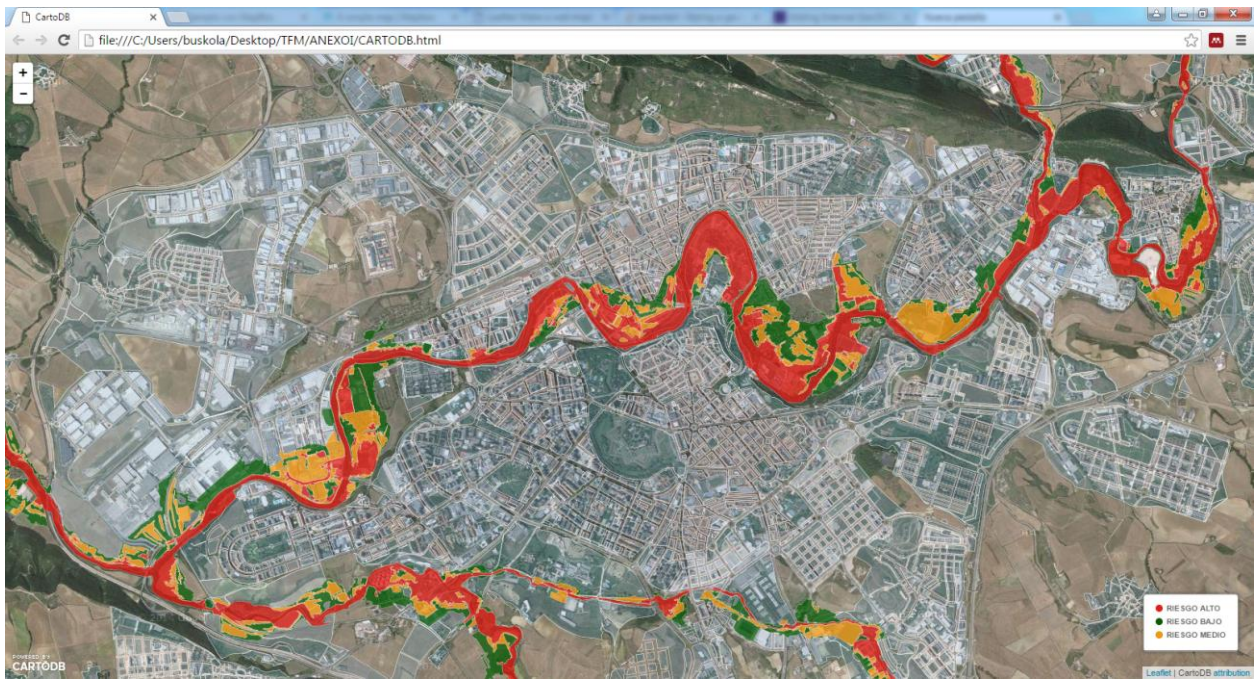


Figura 39. Mapa creado con CartoDB

Por lo que se ha podido ver durante este pequeño ejemplo, esta herramienta permite de forma fácil e intuitiva crear mapas fáciles de integrar dentro de una aplicación web, sin tener que desarrollar grandes códigos para poder llevarlo a cabo. Tampoco hay que olvidar la potencialidad que da la posibilidad de interactuar con PostGIS mediante consultas SQL.

MAPBOX

MapBox, al igual que CartoDB, dispone de editores que permiten modificar primero las capas y estilos para después insertarlos mediante su API, siendo imprescindible darse de alta en su página oficial. En este caso es posible trabajar con el MapBox Studio Classic que es un software de escritorio, aunque también existen el MapBox Studio y el MapBox Editor que trabajan en línea.

Los Studio (tanto el de escritorio como el de línea) sirven para cambiar los estilos que van a tener las capas a la hora de visualizarlas. Se podrá editar tanto los estilos de las capas base (terreno, carreteras, edificios, puentes, bosques…) como las capas que se quieren añadir. También dispone de estilos propios para añadir como por ejemplo 'Comic!', 'Pencil', 'Run, Bike and Hike', 'Mapbox Dark', 'Mapbox Light', 'Pirates Return'…

Para empezar a diseñar el mapa que se espera mostrar en la aplicación web se diferencian dos partes (figura 40):

- Styles: Es lo primero que se debe crear. Hace referencia al estilo que tendrá el mapa, definiendo el estilo de cada uno de los elementos que aparezcan en él. Será posible escoger uno de la lista (con opción a modificarla) o empezar desde cero.
- Sources: son aquellas capas externas que se quieran añadir al mapa. Permite archivos en Shapefile, KML, GeoJSON, GPX, CSV, GeoTIFF, VRT, PostGIS y SQLite. Desde este punto será posible eliminar elementos o atributos, y en el caso de tener los datos en PostGis, realizar consultas SQL.

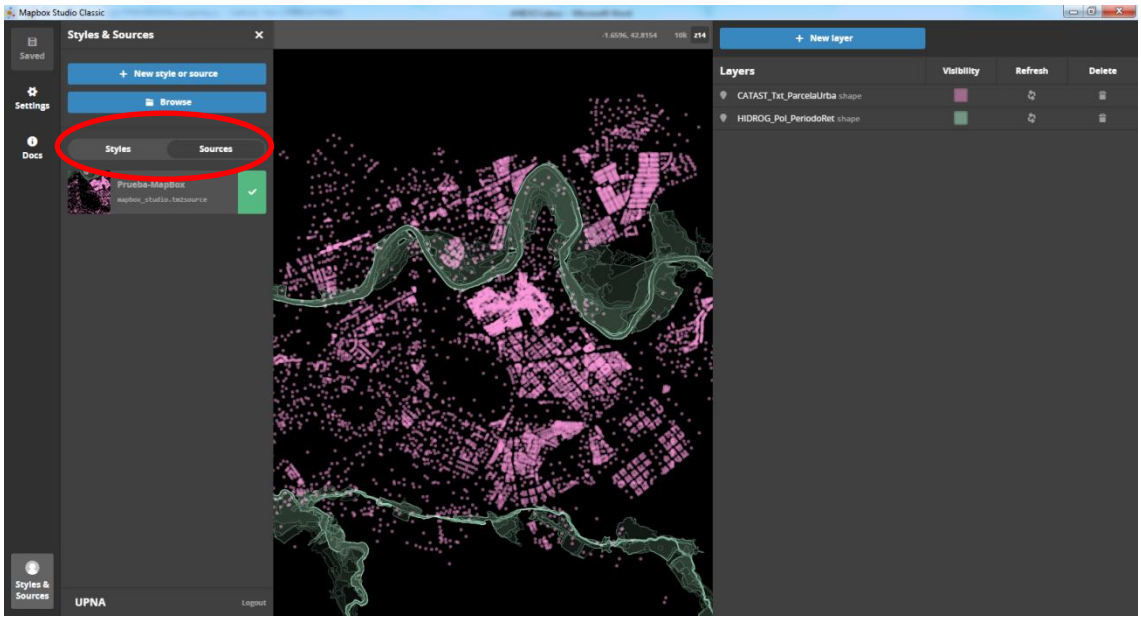


Figura 40. MapBox Studio Classic (software de escritorio)

Cuando se haya cargado y modificado los elementos que aparecerán al gusto del diseñador desde el software de escritorio, es necesario subir lo creado (figura 41) a la cuenta que se tenga creado en MapBox, es decir, al MapBox Studio, guardándolo de esta manera en la nube.

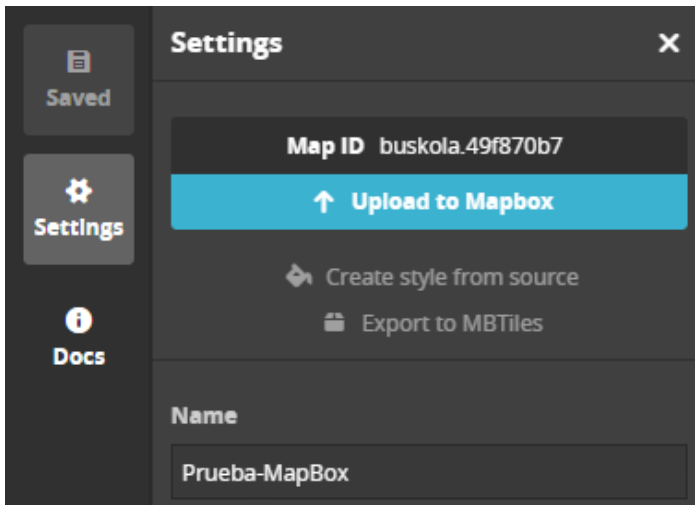


Figura 41. Desplegable de MapBox Studio Classic (en Settings) para subir lo creado a la cuenta de MapBox

En comparación a las herramientas proporcionadas por CartoDB, este MapBox Studio (Classic) no resulta tan intuitiva y la curva de aprendizaje es más baja, no es fácil comprender el funcionamiento de cada opción. Se puede decir que en un principio están más orientadas a diseñadores que a análisis GIS.

En cuanto al Editor que tienen en línea, lo que sería otra de las posibilidades para la carga de datos, se escoge el estilo de la capa de entre los de defecto y es posible añadir capas externas en formatos GeoJSON, KML (no funciona con KMZ), GPX y CSV.

Por lo que se ha podido comprobar durante la realización de esta práctica (mayo de 2016) surgen problemas con los GeoJSON y solamente admite archivos KML que ocupen poco, imposibilitando trabajar con datos de inundación ni qué decir de los de catastro. Por ese motivo se ha continuado con los subidos mediante el MapBox Studio Classic.

Lo siguiente es acceder a la cuenta de MapBox Studio y extraer de la misma los códigos o 'keys' necesarias para conectar mediante la API a lo guardado en la nube: acces token, Style URL y Map ID (figuras 42 y 43). No hará falta este último código si las capas subidas se añaden al estilo.

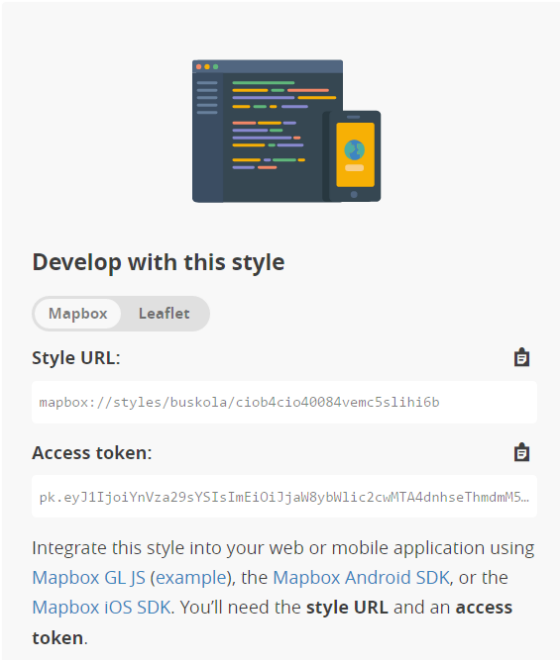


Figura 42. Apartado de MapBox Studio donde aparecen la 'access token' y la 'Style URL' para integrarlos en la API

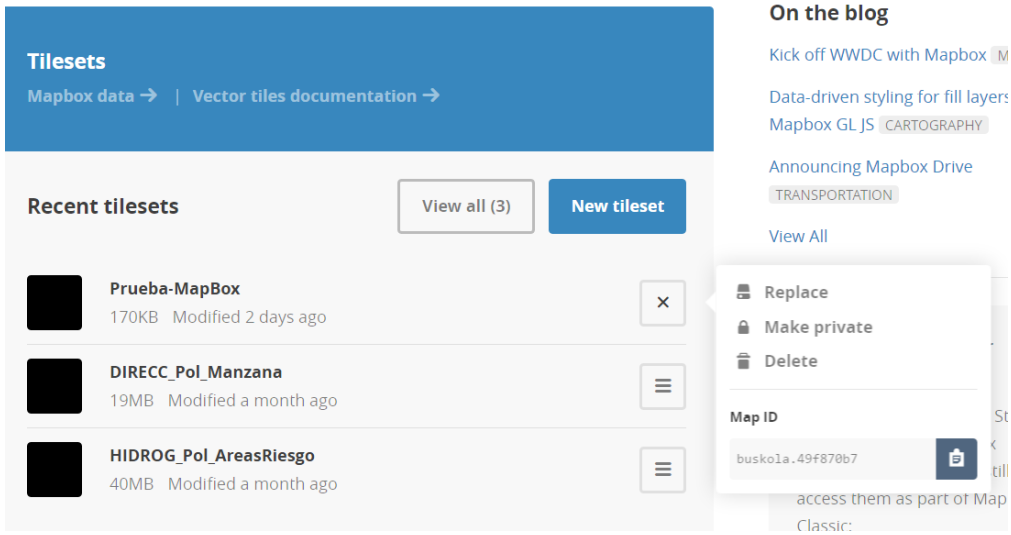
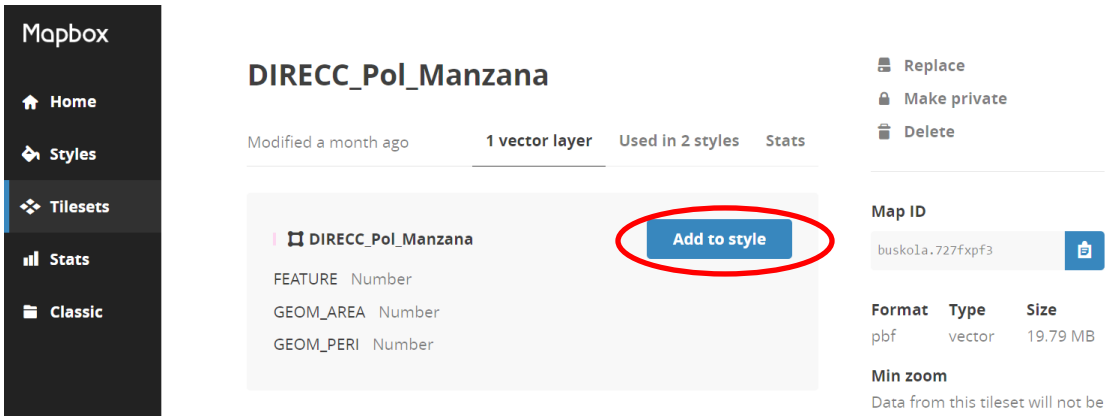


Figura 43. Apartado de MapBox Studio donde aparecen la 'Map ID' para integrarlo en la API



Una vez se han conseguido todos los códigos necesarios para que desde la API se pueda acceder a los estilos y capas guardadas en la nube, lo siguiente ya es desarrollar el código. En este caso la estructura del HTML + JavaScript se ha basado en la librería MapBox js en lugar de MapBox GL JS, y el script contiene tres 4 funcionalidades principales:

- Poner el 'access token' y crear el mapa.
- Añadir estilo, que tiene incorporado la capa vectorial de 'Manzanas' descargada desde el IDENA.
- Añadir capa WMS de los `Periodos de Retorno' conectándose al IDENA.

El código es el siguiente:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset=utf-8 />
    <title>MapBox</title>
    <meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
    <script src='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.js'></script>
    <link href='https://api.mapbox.com/mapbox.js/v2.4.0/mapbox.css' rel='stylesheet' />
    <style>
      body { margin:0; padding:0; }
      #map { position:absolute; top:0; bottom:0; width:100%; }
    </style>
</head>
<body>
    <div id='map'></div>
    <script>
    window.onload = function() {
        L.mapbox.accessToken = 'pk.eyJ1IjoiYnVza29sYSIsImEiOiJjaW8ybWlic2cwMTA4dnhse
ThmdmM5ZDRuIn0.Vbzpt2xLGQCv_xCak02IJQ';
```

```
        var map = L.mapbox.map('map')
            .setView([42.82, -1.65], 15);
        //Integrar estilo creado en Mapbox Studio
        L.mapbox.styleLayer('mapbox://styles/buskola/ciob4cio40084vemc5slihi6b').addTo(map);
        //wms https://www.mapbox.com/mapbox.js/example/v1.0.0/wms/
        var retorno = L.tileLayer.wms('http://idena.navarra.es/ogc/wms', {
            format: 'image/gif',
            transparent: true,
            layers: 'IDENA:HIDROG_Pol_PeriodoRet'
        }).addTo(map);
    }
    </script>
</body>
</html>
```

El resultado obtenido es:

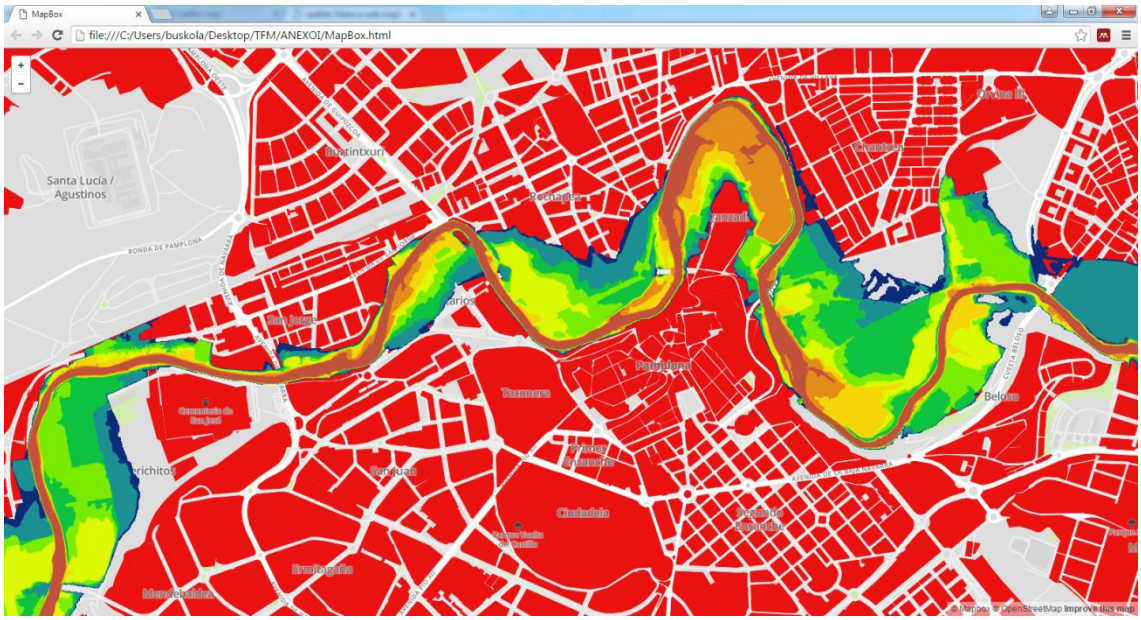


Figura 44. Mapa creado con MapBox

ANEXO II. DESCARGA DE DATOS DESDE LA INFRAESTRUCTURA DE DATOS ESPACIALES DE NAVARRA (IDENA)

La Infraestructura de Datos Espaciales (IDE) de Navarra (IDENA) es la respuesta del Sistema de Información Territorial de Navarra (SITNA) a los requerimientos de INSPIRE (Infrastructure for Spatial Information in Europe) y de la IDEE (Infraestructura de Datos Espaciales de España) para promover y respaldar la armonización, difusión y utilización de datos espaciales (SITNA, 2016a).

IDENA se divide en 4 apartados distintos relacionados entre sí: Buscar, Ver mapas, Descargar y Servicios. Cada uno de ellos aporta distintas funcionalidades que facilitan el acceso a los datos espaciales de Navarra. La dirección URL es:

http://idena.navarra.es/Portal/Inicio

BUSCAR (figura 45)

Como el propio nombre indica, este apartado sirve fundamentalmente para buscar los datos que sean de interés. Existen dos maneras de realizar la búsqueda:

- (1) Introducir nombre clave para realizar la búsqueda en el catálogo de metadatos.
- (2) Hacer búsqueda dependiendo de la organización propietario del recurso, fecha o criterios INSPIRE (divide los datos por temáticas).

Los resultados que se obtengan aparecerán en la ventana central (3), donde será posible acceder a los metadatos y consultar las características del dato. También será posible en algunos de los casos descargar un zip que contenga los archivos Shapefile y el XML de los metadatos .

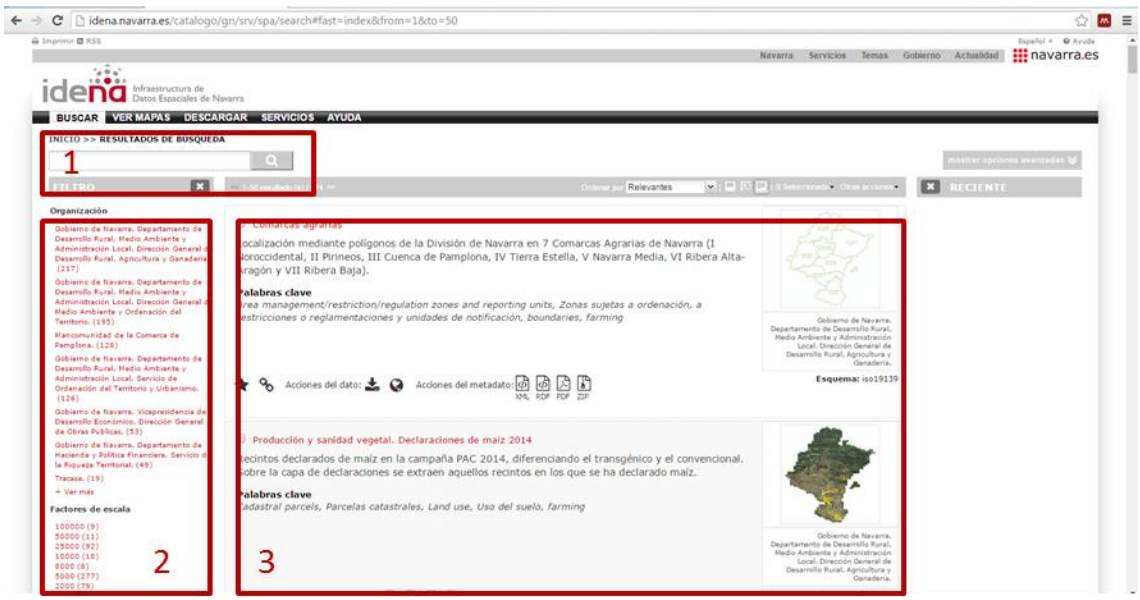


Figura 45. Buscador de IDENA

VER MAPAS (figura 46)

Se refiere al visor del IDENA. Permite visualizar las capas servidas por IDENA para poder observar las características de los datos antes de descargarlos o utilizarlos. Contiene dos desplegables a ambos lados para facilitar la comprensión: herramientas (1) y leyenda (2).

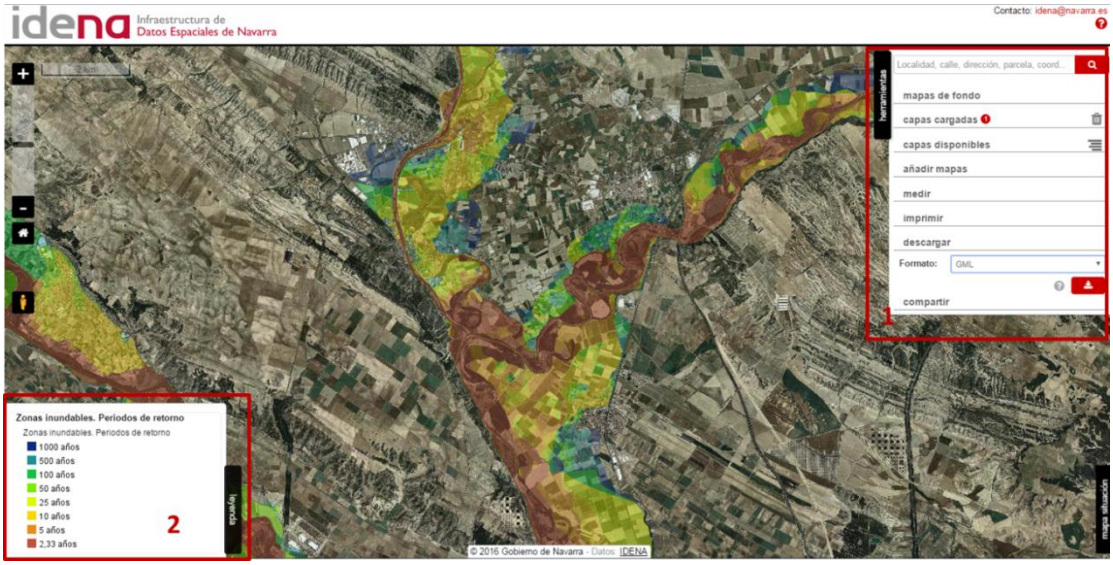


Figura 46. Visor de IDENA

En las herramientas se podrá elegir un mapa base, cargar capas disponibles de IDENA, añadir otras capas de otros servicios web, manipular las capas ya cargadas (superponerlas o cambiar la transparencia) y medir, imprimir, compartir o descargar (en formatos SHP, GML, GeoJSON o KML) los datos que en ese momento aparecen en el visor.

La leyenda explica qué se provee sobre los símbolos y colores que se presentan en el mapa, dividida entre las capas que ya están cargadas.

Para mayor detalle se puede consultar el 'Manual de usuario del visor geográfico de la API SITNA' (SITNA, 2016b)

DESCARGAR (figura 47)

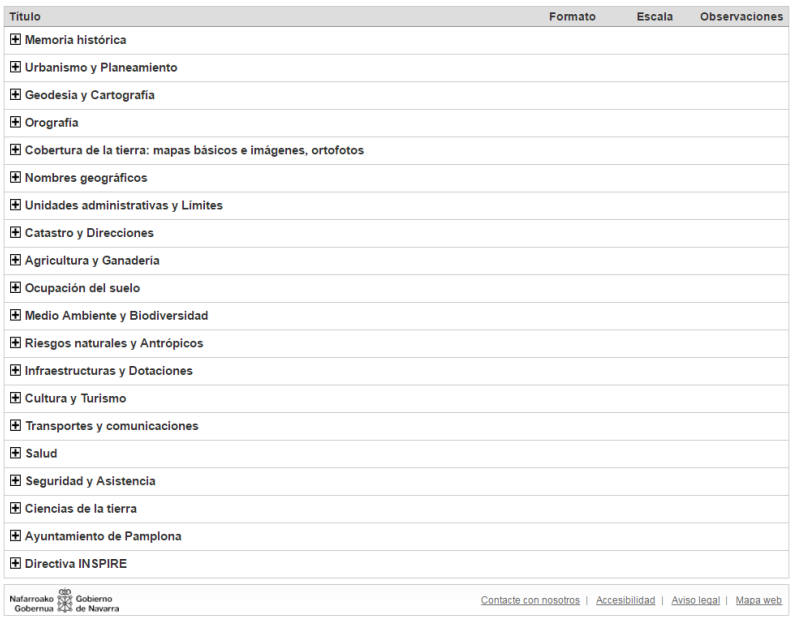


Figura 47. Descargas de IDENA

En este apartado se suministra la información disponible del IDENA en el sistema de referencia EPSG: 25830 (UTM 30N, ETRS89), mayoritariamente en formato SHP. Una vez se haya escogido cuáles son los datos que se quieren descargar y si se quiera trabajar con ellos teniendo los datos de forma física en el ordenador, en esta sección todas las capas están clasificadas por temáticas. Los datos se descargan en un zip que contiene los archivos shapefile.

SERVICIOS (figura 48)

IDENA pone a disposición del público los distintos servicios web requeridos por INSPIRE para que se pueda acceder a los datos espaciales de forma remota mediante clientes ligeros o pesados. Los servicios son:

- Web Map Service (WMS): proporciona acceso a la visualización de la información geográfica.
- Catalogue Service (CSW): permite la publicación y búsqueda de información (metadatos) que describen datos, servicios, aplicaciones y en general todo tipo de recursos.
- Web Feature Service (WFS): permite poder acceder y consultar todos los atributos de un fenómeno (feature) geográfico como un río, una ciudad o un lago, representado en modo vectorial, con una geometría descrita por un conjunto de coordenadas.
- Web Coverage Service (WCS): permite la obtención de datos geoespaciales en forma de "coberturas", es decir, información geográfica espacial digital que representa fenómenos con variación espacio/temporal, de modo que sean útiles para la representación digital de este tipo de información geográfica, como dato de entrada de modelos científicos o para otro tipo de clientes.
- Web Map Tile Service (WMTS): se basa en un modelo de teselas con estructura piramidal que pre-renderiza y fragmenta los datos geográficos a un tamaño de celda concreto para un determinado conjunto de escalas. La ventaja de este servicio respecto al servicio WMS es la velocidad de respuesta.

Figura 48. Servicios web de IDENA

ANEXO III. EDICIÓN DE CAPAS EN QGIS: CREACIÓN DE NUEVAS CAPAS Y MODIFICACIÓN DE LAS YA EXISTENTES

En el caso de que los datos espaciales ya existentes no sean los adecuados para el trabajo que se vaya a realizar puede que convenga modificar esos datos con acciones como borrar elementos, insertar uno nuevo o editar su posición. En otros casos simplemente esos datos no existen, y por lo tanto habrá que crear una nueva capa que después se irá poblando. En este documento se pretende explicar cómo realizar estas acciones en el software GIS de escritorio QGIS Desktop 2.8.9.

Como para poder modificar una capa esa capa tiene que existir, primero se expondrá cómo crear una capa que esté vacía y después las acciones que sirven para modificar cualquier capa, este vacía o con elementos.

CREACIÓN DE UNA NUEVA CAPA VACÍA

Solamente hace falta un paso para poder crear una nueva capa que esté vacía. La herramienta de QGIS que permite hacerlo se encuentra en *Capa -> Crear Capa -> Nueva capa de archivos shape…* (figura 49), que permite crear un nuevo archivo Shapefile donde se encontrará la capa.

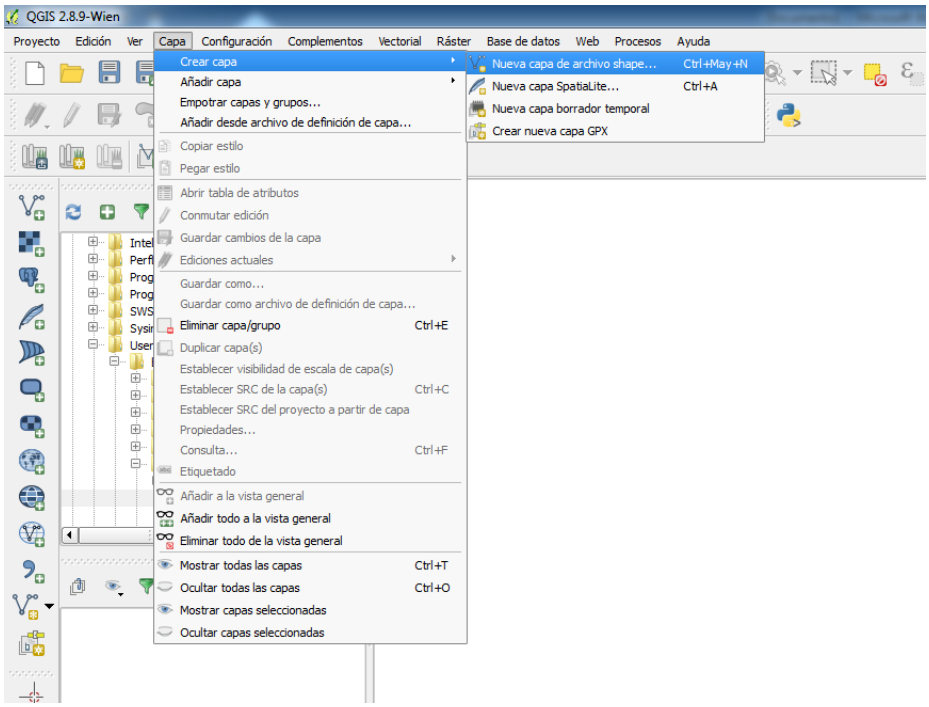


Figura 49. Ubicación en QGIS de la herramienta de creación de capas nuevas

En la ventana que se abre a continuación (figura 50) se deberán rellenar los campos obligatorios:

- Tipo (1): cada capa solamente puede contener elementos de tipo Punto, Línea o Polígono.
- Sistema de referencia (2): en este caso, para trabajar en las localidades de Navarra junto a capas descargadas desde el IDENA, el sistema más adecuado será el ETRS89/UTM zona 30N (EPSG: 25830).
- Nuevo atributo (3): por defecto todas las capas tienen un atributo de nombre 'id', pero se pueden añadir tantos atributos como se quieran. Para poder añadir habrá que asignar un nombre y el tipo de dato (texto, integer, float o fecha) con su anchura y precisión.
- Aceptar (4): al aceptar se abrirá una nueva ventana para elegir la ubicación del nuevo archivo.

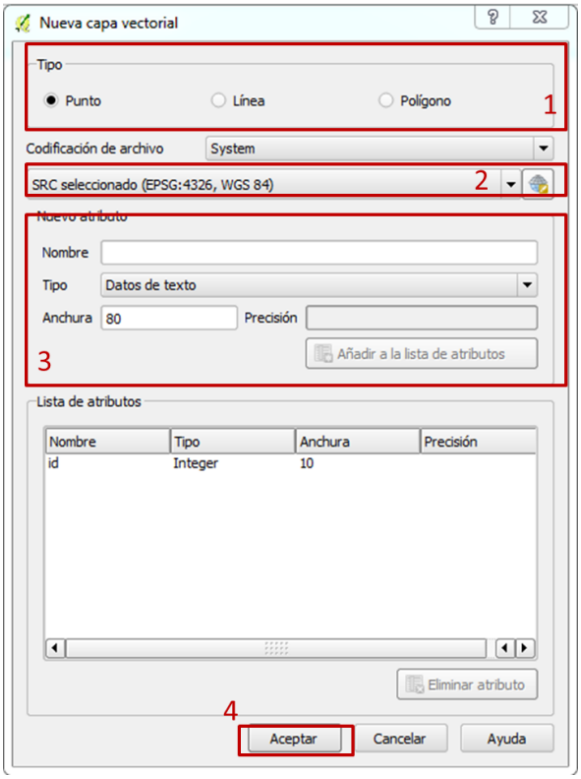


Figura 50. Herramienta de QGIS para crear nueva capa vacía

MODIFICAR CAPAS

Para modificar una capa en QGIS, este vacía o contenga elementos, están disponibles dos barras de herramientas: 'Digitalización' y 'Digitalización Avanzada'. En el caso de que su visualización no esté activada, se podrá activar siguiendo *Ver -> Barra de herramientas -> Digitalización/Digitalización avanzada* (figura 51).

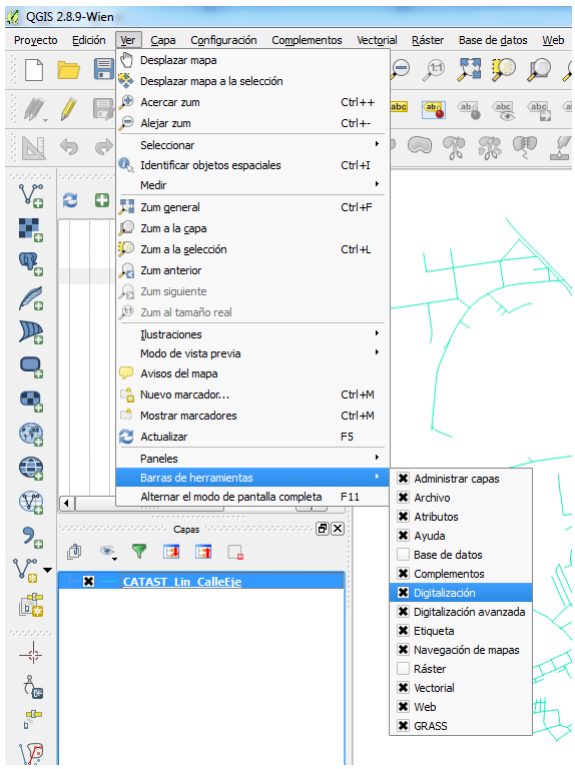


Figura 51. Ubicación en QGIS de las barras de herramientas

Dentro de estas barras se encuentran todas las herramientas disponibles para modificar las capas. Tal y como indican sus nombres, las acciones básicas se encuentra en la barra 'Digitalización' (figura 52) y las más avanzadas en 'Digitalización Avanzada' (figura 53).

Figura 52. Barra de herramientas de 'Digitalización'

Figura 53. Barra de herramientas de 'Digitalización Avanzada'

Para empezar a editar cualquier capa, el primer paso es tener seleccionada la capa y conmutar la edición (1). Con esta acción se activan todas las herramientas que se puedan usar en ese momento (dependiendo del tipo de capa se activarán unas u otras) y en la visualización se visualizan los nodos (figura 54).

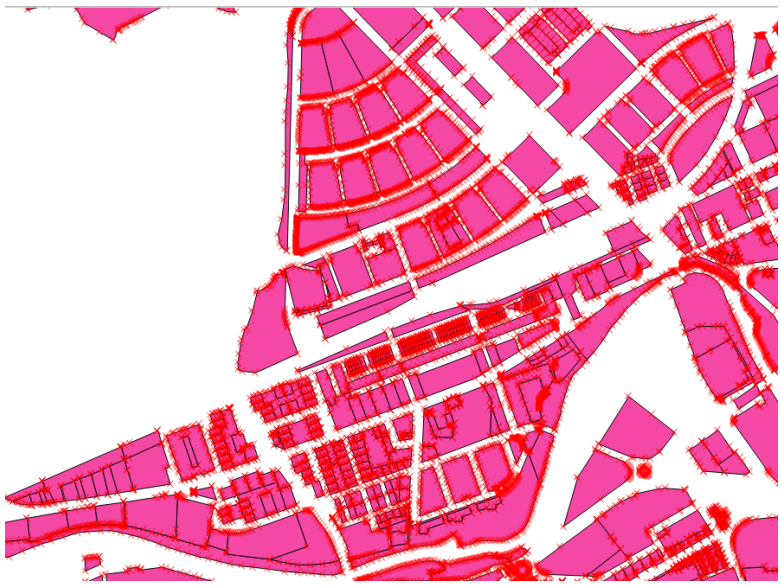


Figura 54. Visualización en QGIS de una capa de tipo polígono cuando se conmuta la edición: nodos representados con cruces rojas

Las posibilidades de edición son muchas, pero las más útiles son:

- (2): Añadir objeto espacial. Con el botón izquierdo del ratón se insertan los nodos y con el derecho se finaliza la acción. Al finalizar se abre una ventana para introducir los valores de los campos.
- (3): Borrar lo seleccionado. Para poder hacerlo primero se deberán seleccionar los elementos .
- (4): Mover objeto espacial y Herramienta de nodos. Son las acciones básicas para editar los elementos.
- (5): Son las acciones avanzadas de edición. Cambian dependiendo del tipo de capa, pero todas son parecidas: rotar, simplificar, añadir parte, eliminar parte, remodelas objetos, desplazar curvas, dividir objetos, dividir partes…

Para terminar con lo que se haya realizado, se vuelve a pulsar el botón de conmutar la edición (1) y se elige si se va a guardar o no lo que se ha cambiado.

MAPA BASE

Puede resultar de gran ayuda disponer de un mapa base como pueden ser los suministrados por Open Street Map, Google Maps o Bing Maps para poder goelocalizar correctamente los elementos del mapa que se quieran modificar.

Existe un plugin en QGIS denominado 'OpenLayers plugin' que se podrá instalar en *Complementos -> Administrar e instalar complementos…* (figura 55) En la ventana que abre se busca el complemento y se pulsa el botón 'Instalar complemento'.

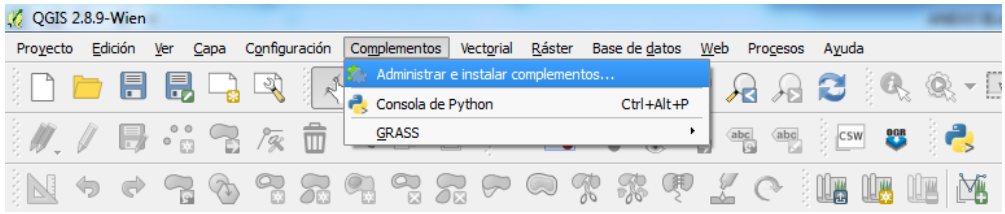


Figura 55. Ubicación en QGIS del administrador de complementos

Se accede a la herramienta en *Web -> OpenLayers plugin,* donde se puede escoger entre distintos proveedores los productos que se pueden poner como mapa base (figura 56). De este modo todas las capas cargadas se superponen encima del mapa base facilitando la identificación de elementos que faltan o que por alguna razón tienen que ser modificadas.

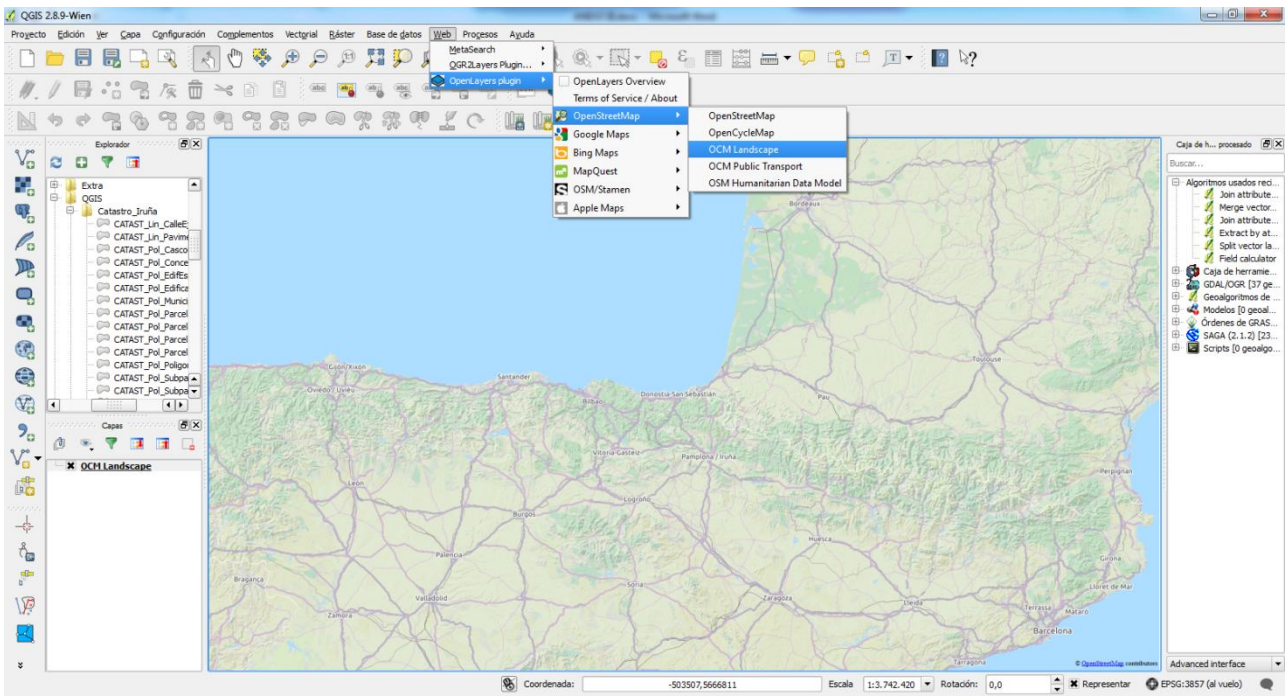


Figura 56. QGIS con mapa base de Open Street Map cargada con el plugin de OpenLayers

ANEXO IV. PRE-PROCESAMIENTO DE DATOS

El pre-procesamiento empieza con los archivos Shapefile descargados y consiste en eliminar, unir o modificar las tablas y sus atributos para que se puedan integrar en la base de datos. El anexo se divide en cuatro partes: en las dos primeras se expone cómo tratar con los datos descargados, después se explica cómo cargar capas a la base de datos y finalmente una pequeña introducción de cómo aplicar operaciones SQL para crear nuevas tablas en PostGIS. El objetivo es que al ejecutar todos estos procesos la base de datos este lista y preparada para empezar el geoprocesamiento y calcular elementos afectados.

DATOS REFERENTES A TODA LA COMUNIDAD FORAL DE NAVARRA

Son datos que se puedan descargar desde el IDENA que ocupan toda la totalidad del territorio. La idea es crear capas de elementos para toda Navarra que se utilizarán en todos los municipios y que se recortarán con los límites de cada municipio a la hora de analizar municipios concretos.

Lo que se ha decidido es crear una capa para cada temática y tipo de geometría (puntos, líneas y polígonos) intentando englobar en lo posible para no tener demasiadas tablas y tener ordenada la base de datos.

- Periodos de retorno
- Ejes de carreteras generales
- Puntos de interés: centros educativos, centros de salud, instalaciones deportivas, recursos turísticos, alojamientos turísticos, buzones, casas consistoriales, depuradoras, centrales, subestaciones y centros de transformación eléctrica… que se puedan unir en una única tabla de puntos. De esta manera se mostrarán todos a la vez en la aplicación web simplificando el código y aumentando la eficacia.

Sin embargo no todos los puntos de interés que se pretenden implementar en la aplicación existen o están disponibles en IDENA u otra IDE revisada (servicios web estatales), como es el caso de los polígonos industriales o transformadores eléctricos. La solución pasa por digitalizarlos manualmente (Anexo III) partiendo de alguna fuente de información como puede ser el caso de los polígonos industriales con:

http://www.gestiondepoligonos.com/poligonos-industriales-Navarra

También se pueden encontrar casos donde solamente esté disponible el servicio de visualización, WMS, y no exista el de descarga, WFS, y que por lo tanto una de las posibles soluciones pasa por digitalizar los puntos en QGIS a mano poniendo la capa WMS como base. Este es el caso de las localizaciones de las industrias agroalimentarias, que en el IDENA la capa no es de acceso público y por tanto su utilización está limitada a los usuarios autorizados y el MAGRAMA solamente lo sirve mediante el servicio WMS.

Para crear una única tabla con todos los puntos será imprescindible que contengan los mismos atributos. En este caso se ha decidido emplear tres atributos y eliminar el resto:

- tipo: texto de longitud 50. Servirá para identificar qué tipo de punto de interés es.
- nombre: texto de longitud 50. Identificará mediante un nombre a qué elemento en concreto se refiere.
- otro: texto de longitud 50. En el caso de que se vea necesario, este campo ayudará a mostrar alguna información de ayuda.

Los pasos a seguir en QGIS son modificar independientemente cada capa y unirlas todas al final (figura 57):

- Poner en edición la tabla (1)
- Crear nuevas columnas (2)
- Rellenar las columnas creadas (3)
- Eliminar columnas no deseadas (4)
- Finalizar edición (1)

- Unir todas las tablas en una: *Vectorial -> Herramientas de gestión de datos -> Combinar archivos shape en uno…* (figura 58)

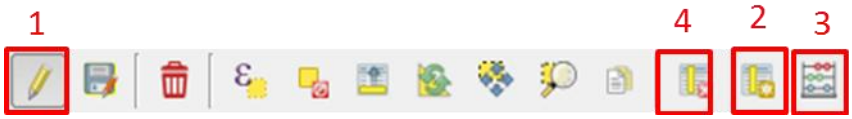


Figura 57. Barra de herramientas de las tablas en QGIS

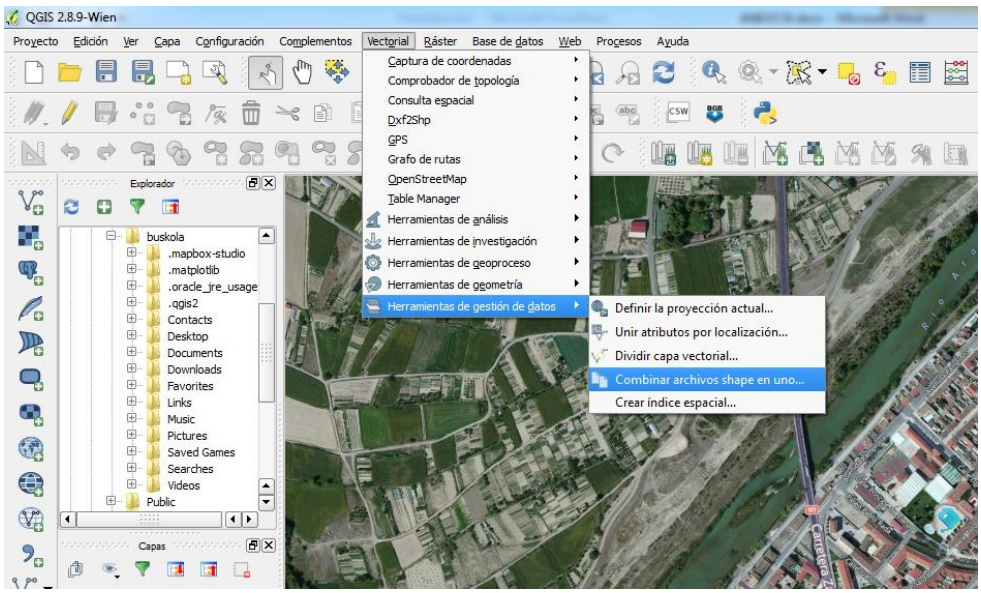


Figura 58. Ubicación en QGIS de la herramienta 'Combinar archivos shape en uno…'

A la hora de rellenar los atributos el esquema a seguir es la siguiente: en 'tipo' se escribe de qué tipo es, en 'nombre' qué atributo del Shapefile original se utilizará para rellenarla y en 'otro' otra vez qué atributo del Shapefile original se utilizará para rellenarla. De este modo se irá rellenando una tabla como la tabla 5 para saber cómo se ha diseñado esta capa hasta el momento y saber cómo rellenar en el caso de querer introducir nuevos puntos.

Tabla 5. Ejemplo de puntos de interés y los valores a utilizar para rellenar campos: entre ' ' campos a rellenar a mano y en mayúsculas campos de la tabla original que pueden ser copiados

tipo	nombre	otro
'Buzon'	DIRECCION	
'Alojamiento Turistico'	RECTUR	TIPO
'Casa Consistorial'	MUNICIPIO	
'Centro Comercial'	EMPRESA	TIPO
'Centro de Educacion'	CENEDUCA	TIPO
'Instalacion Deportiva'	INSTDEP	
'Recurso Turistico'	RECTUR	TIPO
'Centro de Salud Primaria'	CENSAN	
'Depuradora'	PLANTA	CAUCE
'Poligono Industrial'	'Nombre'	
'Industria Agroalimentaria'	'Centro'	'Familia'[4]
…		

[4] Consultable mediante la operación GetFeatureInfo que proporciona el servicio WMS.

En cuanto a la capa de los periodos de retorno, la tabla de atributos consta de campos como el tipo de feature, la cuenca, el área o el perímetro, pero lo que realmente interesa es el periodo de retorno. Lo que pasa es que se trata de un campo de tipo cadena texto y son más fáciles de ordenar y hacer referencia las de números enteros, así que se editará la tabla con las herramientas de la figura 57:

- Poner en edición la tabla (1)
- Crear nueva columna (2): nombre 'retorno' y tipo 'integer' de longitud 3.
- Rellenar las columnas creadas (3):

```
CASE WHEN PRETORNO='10 años' THEN 10
WHEN PRETORNO='100 años' THEN 100
WHEN PRETORNO='1000 años' THEN 999
WHEN PRETORNO='2,33 años' THEN 2
WHEN PRETORNO='25 años' THEN 25
WHEN PRETORNO='5 años' THEN 5
WHEN PRETORNO='50 años' THEN 50
WHEN PRETORNO='500 años' THEN 500
END
```

- Eliminar columnas no deseadas (4): todas menos la que se acaba de crear.
- Finalizar edición (1)

Las carreteras generales no necesitan ningún cambio, así que se pueden importar directamente a la base de datos.

DATOS REFERRENTES A CADA UNO DE LOS MUNICIPIOS A ESTUDIAR

Son principalmente los datos del catastro. La descargar del catastro desde el IDENA solamente es posible hacerlo por municipios, y es obligatorio bajarse todas las tablas en su conjunto, donde se encuentran:

- Ejes de las calles (líneas)
- Pavimento (líneas)
- Casco urbano (polígono)
- Concejo (polígono)
- Edificios Especiales (polígonos)
- Edificación (polígonos)
- Municipio (polígonos)
- Parcelas Mixtas (polígonos)
- Parcelas Rústicas (polígonos)
- Parcelas Urbanas (polígonos)
- Parcelas Viarias (polígonos)
- Polígonos (polígonos)
- Subparcelas Mixtas (polígonos)
- Subparcelas Rústicas (polígonos)
- Subparcelas Urbanas (polígonos)
- Calles (puntos)
- Alturas de edificios (puntos)
- Parcelas Mixtas (puntos)
- Parcelas Rústicas (puntos)
- Parcelas Urbanas (puntos)
- Polígonos (puntos)

- Portales (puntos)
- Subparcelas (puntos)

Naturalmente no son necesarias todas las capas, y por lo tanto lo primero es identificar cuáles son las que se utilizarán para eliminar el resto y limpiar así la base de datos. Para este trabajo en concreto se ha decidido quedarse con las capas del contorno (CATAST_Pol_Municipio), ejes de calle (CATAST_Lin_CalleEje), parcelas urbanas (CATAST_Pol_ParcelaUrba), parcelas rústicas (CATAST_Pol_ParcelaRusti) y portales (CATAST_Txt_Portal).

El contorno servirá para cortar las capas de los datos de toda la Comunidad Foral de Navarra y así conseguir reducir el coste computacional en los siguientes procesos, ya que al ser menor la cantidad de elementos a procesar disminuirá la complejidad del problema.

Los ejes de las calles serán para analizar cuáles son las calles afectadas y en qué punto se cortarían, por supuesto para cada periodo de retorno.

Las parcelas, tanto urbanas como rústicas, ejercerán casi como variables que ocupen prácticamente la totalidad del territorio y que facilitará de alguna manera conocer qué es lo que queda debajo de las manchas. En vez de clasificar la parcela completa como elemento afectado se podrá conocer qué áreas son las que realmente están en peligro con la herramienta de geoprocesamiento adecuada.

Los portales no se representarán en el visor pero se ocuparán de asignarles calles y número de portales a las parcelas para que sea ésta la información que proporcionen en el visor. Es decir, ya que los portales contienen campos más significativos que las parcelas, se relacionarán ambas tablas para que cualquier parcela tenga asignada una calle y un número de portal.

IMPORTAR LOS DATOS A LA BASE DE DATOS

El tener los datos en una base de datos aporta la seguridad de que siempre que el servidor esté disponible se podrá acceder a ellas desde cualquier ordenador con total seguridad. También es importante para poder publicar las capas y poder cumplir así con los estándares OGC para *Web Mapping.*

Además la posibilidad de interactuar ente ellos que tienen QGIS y PostGIS mediante la importación y exportación de tablas facilita un flujo de trabajo continuo y dinámico. Esto ayuda a que en cualquier momento sea posible modificar las tablas creadas para poder introducir nuevos elementos o eliminar los que ya no interesan. Por ejemplo si en un municipio se tiene que introducir un elemento nuevo (por ejemplo las plazas) se conecta desde QGIS al PostGIS, se carga la capa de puntos de interés, se modifica introduciendo los valores de la tabla 5 creando un nuevo tipo ‘Plaza’ y se guarda la capa de nuevo en PostGIS.

La base de datos espacial PostGIS, al estar integrado en PostgreSQL, se administra y se gestiona mediante la herramienta pgAdmin III. Este entorno gráfico de escritorio permite conectarse a cualquier base de datos PostGIS que esté ubicado en un servidor y ejecutar acciones sobre ella.

Para conectarse es suficiente con acceder a *File -> Add server…* (o pulsar el icono del enchufe) y rellenar las opciones que aparecen en la ventana para registrar ese nuevo servidor (figura 59) con el nombre de la conexión, la dirección del servidor, el puerto y el nombre de usuario y contraseña del servidor (el mismo que se haya escogido a la hora de su instalación).

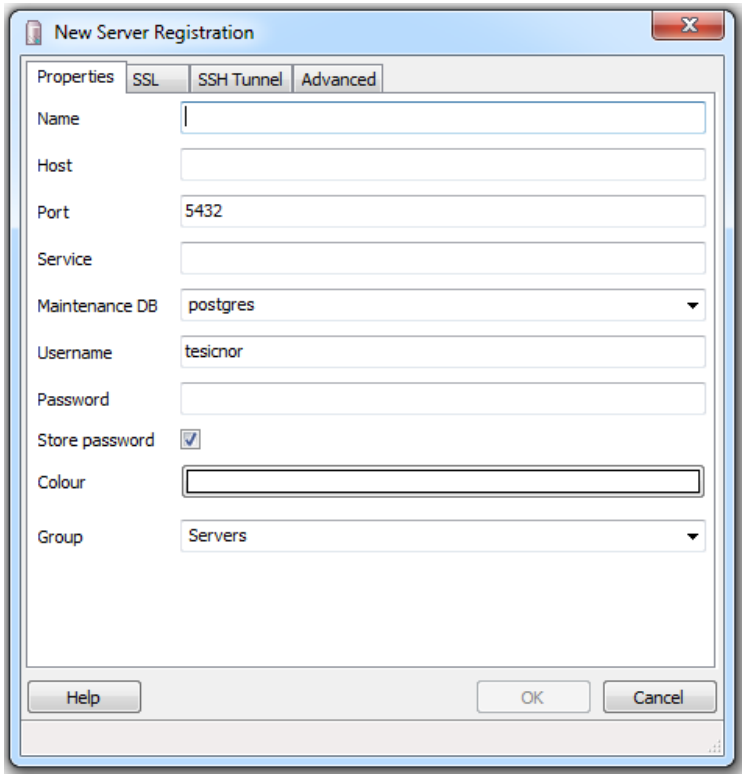


Figura 59. Opciones de pgAdmin III para registrar un nuevo servidor

La estructura cuando se accede a los 'Databases' de un servidor es la siguiente:

- Base de datos
- Esquemas
- Tablas

Si el servidor al que se conecta contiene PostGIS, bajo esa conexión siempre aparecerán creadas por defecto ciertas bases de datos, entre ellas la denominada 'template_postgis'. Siempre que se quiera crear una nueva base de datos espacial será ésta la plantilla que se debe utilizar y garantizar así el componente espacial. Entonces, para crear una nueva base de datos se hace click derecho sobre 'Databases' y *New Database…* A continuación se introduce el nombre y el propietario y en la pestaña 'Definition' se escoge la plantilla espacial como 'Template'. Tras haberla creado conviene asegurarse que dentro del esquema 'Public' se ha creado una tabla por defecto con la información sobre los sistemas de proyección espacial.

Con esto ya se crea una base de datos espacial donde se pueden almacenar tablas dentro de esquemas. En estos casos lo más recomendado suele ser trabajar con pocas bases de datos pero que contengan muchas esquemas. Por ese motivo la base de datos se estructurará de la siguiente forma (figura 60): una base de datos para todos los datos que se dividirá en esquemas, en el esquema por defecto, denominado 'public', se almacenan los datos referentes a toda Navarra y se crea un nuevo esquema para cada municipio interesado donde se incluirán por un lado las tablas del catastro y por otro los cortes de los [periodos; carreteras; puntos interés] con el contorno.

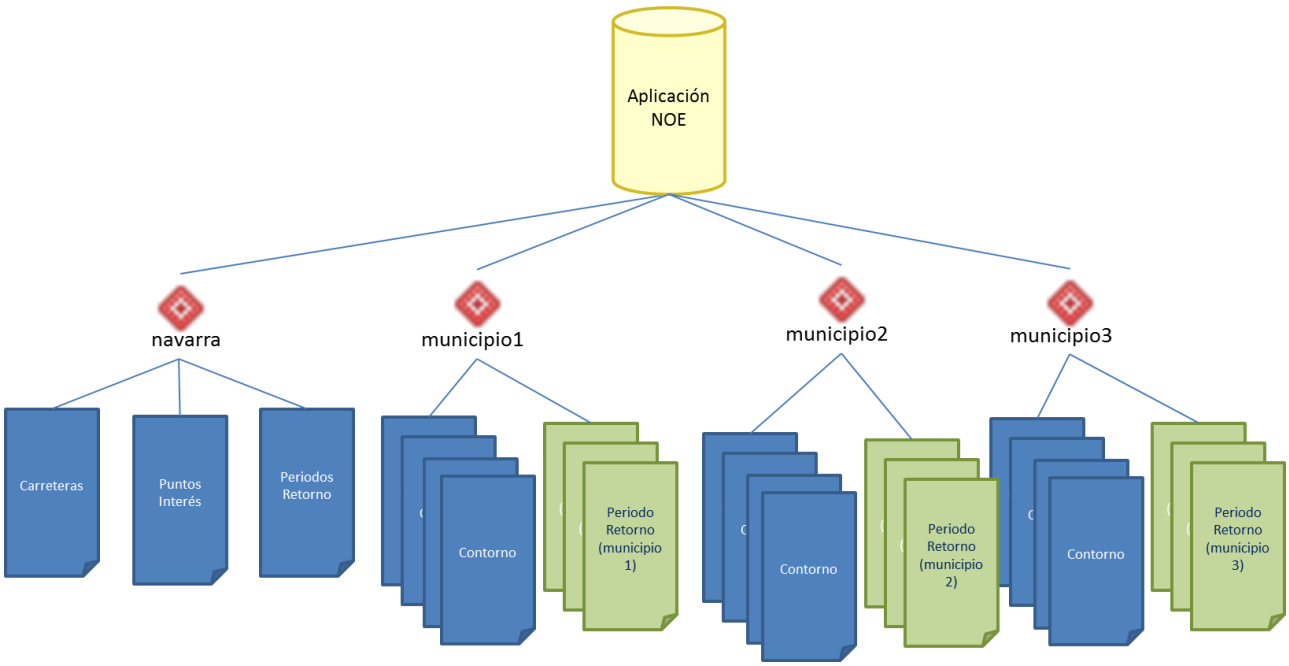


Figura 60. Estructura a crear en PostGIS

Para crear nuevos esquemas se debe abrir el árbol de la base de datos y con el botón derecho del ratón hacer click en 'Schemas' y *New Schema…* Se recomienda que los nombres de los esquemas sean los de los municipios y en minúsculas.

Cuando ya está preparada la estructura de la base de datos, solamente queda cargar las capas. Para poder importar las tablas a PostGIS existen dos maneras:

1. El plugin de pgAdmin III 'PostGIS Shapefile and DBF loader'
2. Conectarse a la base de datos desde QGIS y exportar lo modificado

Para aquellas tablas que no necesiten ningún tipo de edición (las referentes al catastro y ejes de carreteras generales) la manera más cómoda de hacerlo es mediante el plugin. Este plugin se instala junto al PostGIS y es posible acceder a ella desde la barra de herramientas de pgAdmin III en 'Plugins' (figura 61). Solamente hará falta escoger el Shapefile que se quiera incluir y escoger el esquema donde se vaya a importar y la id del sistema de referencia (SRID) que en la mayoría de los casos en datos descargados del IDENA será el 25830.

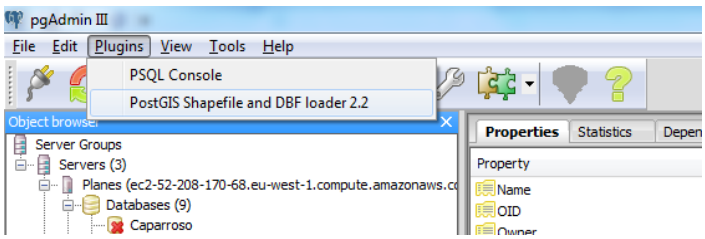


Figura 61. Ubicación del plugin en pgAdmin III

En el caso de que primero se tengan que modificar las capas, puede resultar más cómodo importarlos directamente desde QGIS (también se podría primero guardar las modificaciones en Shapefile y después ejecutar el plugin anterior). Este sería el caso de los periodos de retorno y los puntos de interés. Tras haber modificado las tablas en QGIS se podrían exportar a PostGIS mediante herramientas propias.

El primer paso es conectarse a la base de datos. Para ello se pulsa el botón  y en la ventana que se abre se conecta a una conexión ya creada. Si todavía no se ha creado una nueva conexión habrá que crear

una y rellenar las opciones (figura 62) con: el nombre, dirección del servidor, el puerto, la base de datos a la que se quiere acceder en esa dirección y usuario y contraseña de la base de datos.

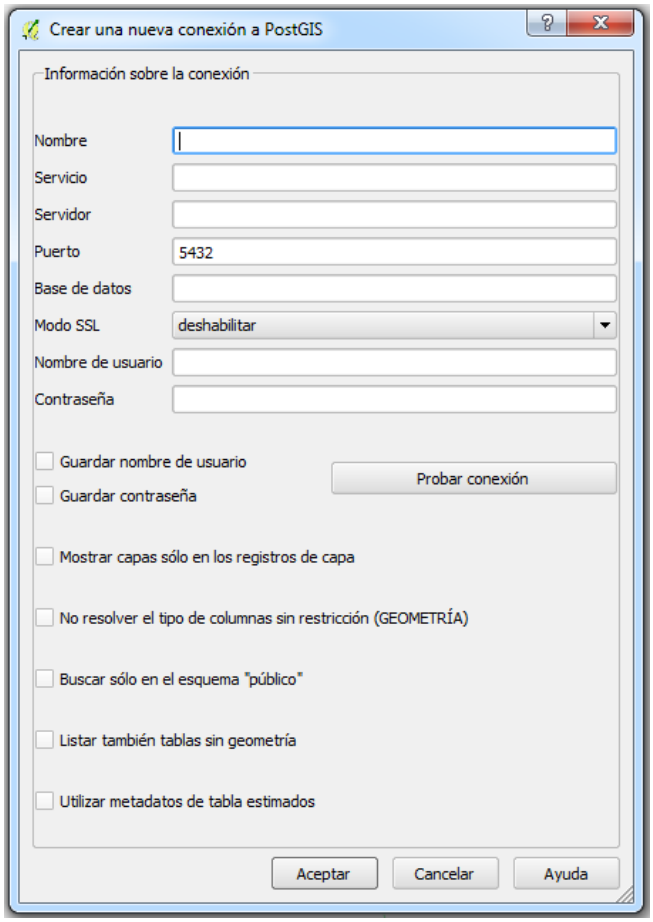


Figura 62. Opciones para crear una nueva conexión a PostGIS desde QGIS

Tras la conexión se accede a la base de datos mediante la herramienta 'Administrador de BBDD' (figura 63). Esta herramienta permite importar y exportar tablas entre el proyecto abierto en QGIS y la base de datos de PostGIS o Spatialite al que se tenga conexión. También permite crear, borrar y actualizar las tablas y esquemas de las distintas bases de datos.

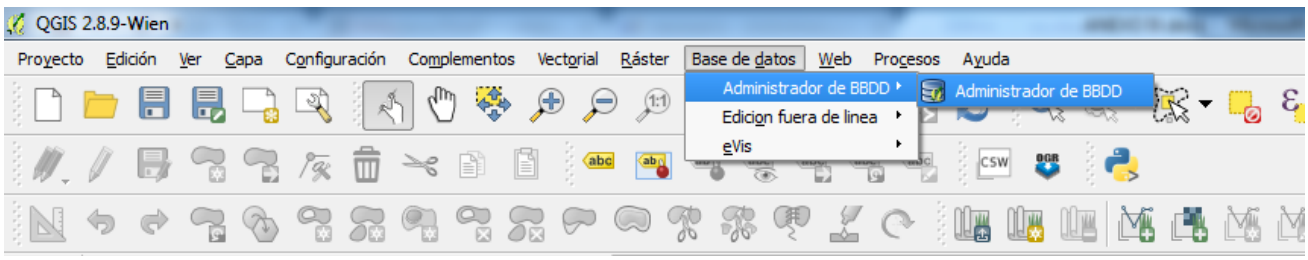


Figura 63. Ubicación del 'Administrador de BBDD' en QGIS

Para poder almacenar una tabla que todavía no existe es suficiente con pulsar el botón  de 'importar capa/archivo' y rellenar las opciones (figura 64) seleccionando la capa que se quiere importar, en qué esquema almacenar y el nombre de la tabla.

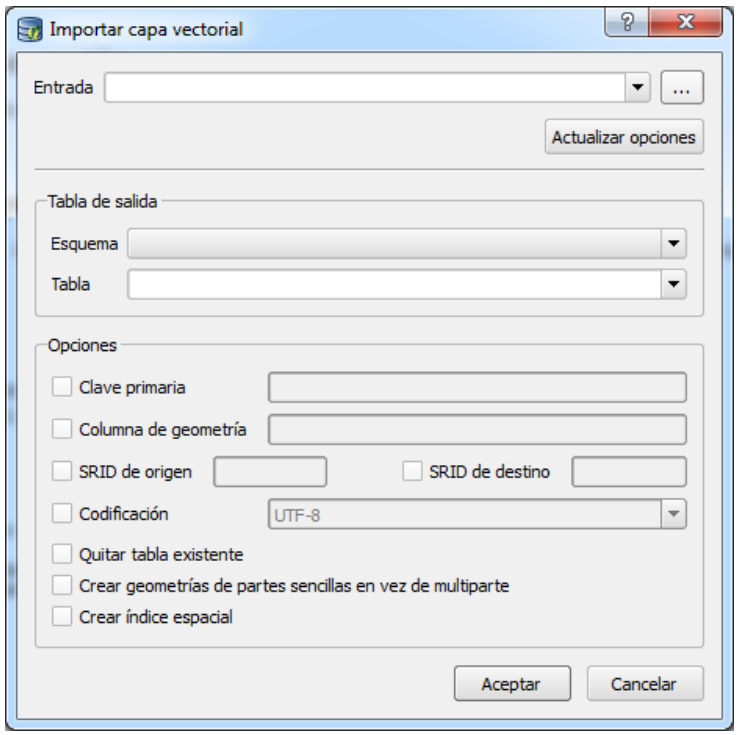


Figura 64. Opciones para la importación de una capa vectorial en PostGIS desde QGIS

OPERACIONES ESPACIALES BÁSICAS CON CONSULTAS SQL

El tener instalado una base de datos espacial como PostGIS posibilita ejecutar operaciones espaciales utilizando las tablas cargadas sin necesidad de trabajar con un software GIS de escritorio (PostGIS, 2016). Esta funcionalidad facilita la automatización del trabajo, y aunque se puede considerar como geoproceso más que pre-proceso, ésta ejecución se incluye en este apartado para que al finalizarla la base de datos esté lista para utilizarla.

Todas las funcionalidades de PostGIS también existen en QGIS, y por lo tanto se puede elegir cualquiera de las dos opciones. La ventaja que tiene QGIS es que no necesita códigos y todo lo que ejecuta se puede representar gráficamente facilitando la comprobación, y además son más las herramientas y más complejas. Pero las consultas SQL permiten automatizar el proceso sin tener que estar importando y exportando tablas a un software GIS de escritorio, permitiendo también crear vistas en vez de tablas para conseguir mayor integridad de los datos (en este caso no se ha utilizado debido al coste en tiempo que implican en pasos posteriores).

El objetivo de este apartado es crear las últimas tablas que faltan para completar la estructura de la base de datos (figura 60): los datos referentes a toda Navarra cortadas por el contorno del municipio.

Para empezar a ejecutar consultas primero se accede al asistente pulsando el botón  en pgAdmin III. En la ventana que se abre (figura 65) es importante comprobar que se ha conectado a la base de datos adecuada en la parte superior central (1). Consta de dos grandes partes: en una se escribe el código (2) y al ejecutarse los resultados aparecerán en el otro (3).

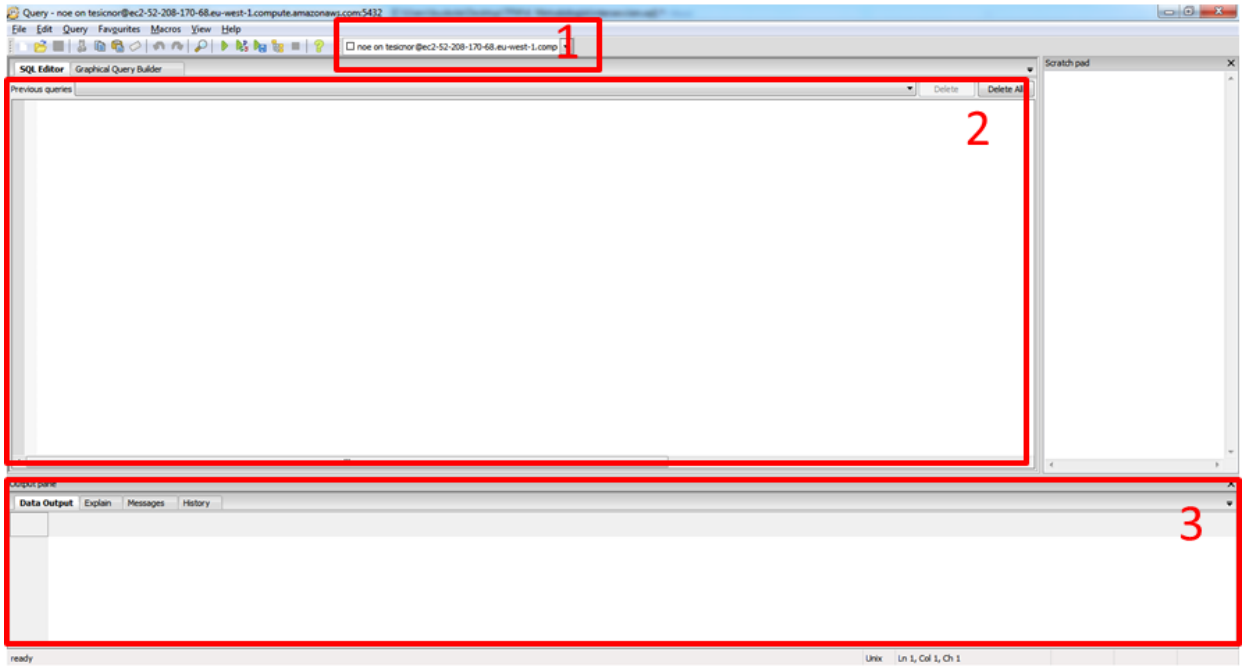


Figura 65. Ventana SQL de PostGIS

Para evitar tener que indicar el nombre del esquema en todas las consultas que se harán facilita el trabajo ejecutar el siguiente comando SQL (se escribe en la ventana SQL y lo ejecutamos con F5).

SET search_path=**nombre_municipio**,public;

En este caso, además del esquema del municipio, se le indica el esquema public ya que en él se encuentran las funciones espaciales de PostGIS que se van a necesitar, además de las tablas almacenadas anteriormente allí.

La función para conocer la lista de elementos que se encuentra dentro del contorno del municipio es el *ST_Intersection (geometry, geometry)* que devuelve una columna de tipo 'geometry'. Esto conlleva a que está función formara parte de las columnas de la tabla de salida, que será la que se utilizará para georreferenciar. No hay que olvidar que solamente se necesitan las tuplas que cumplan la coindición de que se intersecten, por lo que se debe incluir una condición de *ST_Intersects (geometry, geometry)* que devolverá un booleano. El código para ejecutar la operación en las tres capas es la siguiente:

SELECT **columnas,** ST_Intersection(**tabla1**.geom, **tabla2**.geom) FROM **tablas** WHERE ST_Intersects (**tabla1**.geom, **tabla2**.geom)**;**

Lo último será crear una tabla con cada una de las consultas SQL mediante el código *CREATE TABLE* ***nombre_de_la_tabla*** *AS…* (en el caso de trabajar con vistas la función cambia a *CREATE VIEW*). En conclusión, el código completo de la consulta espacial SQL para conseguir el objetivo es el siguiente, siempre y cuando no se hayan cambiado los nombres por defecto de las distintas capas:

```
SET SEARCH_PATH=nombre_municipio, public;

CREATE TABLE pre_carreteras_generales
AS
SELECT c.gid, c.dcarretera, c.carretera, ST_Intersection(c.geom, m.geom) geom
FROM infrae_lin_ctraeje c, catast_pol_municipio m
WHERE ST_Intersects(c.geom, m.geom);

CREATE TABLE pre_periodos_retorno
AS
SELECT r.gid, retorno, ST_Intersection(r.geom, m.geom) geom
FROM hidrog_pol_periodoret r, catast_pol_municipio m
WHERE ST_Intersects(r.geom, m.geom);
```

```
CREATE TABLE pre_puntos_interes
AS
SELECT p.id, tipo, nombre, otro, ST_Intersection(p.geom, m.geom) geom
FROM puntos_interes p, catast_pol_municipio m
WHERE ST_Intersects(p.geom, m.geom);
```

Si se decide que es más fácil trabajar con QGIS existe una herramienta equivalente a la *ST_Intersection* llamado *Clip,* donde se introduce por una parte la capa a recortar (cualquiera de las tres) y la capa de corte que será el contorno. El inconveniente de esta opción es que primero se tendría que conectar a la base de datos desde QGIS, acceder a las capas, hacer una a una la operación y volver a importar cada resultado una a una a la base de datos. Además, trabajar con QGIS impide poder crear vistas.

Con el fin de ordenar un poco más la base de datos, cambiando de nombre las tablas del catastro y dejando solamente las columnas necesarias, también se puede ejecutar la siguiente consulta SQL:

```
CREATE TABLE pre_calles
AS
SELECT gid, via, geom
FROM catast_lin_calleeje;
DROP TABLE catast_lin_calleeje;

CREATE TABLE pre_contorno
AS
SELECT gid, municipio, geom
FROM catast_pol_municipio;
DROP TABLE catast_pol_municipio;

CREATE TABLE pre_parcelas_r
AS
SELECT gid, refcat, geom
FROM catast_pol_parcelarusti;
DROP TABLE catast_pol_parcelarusti;

CREATE TABLE pre_parcelas_u
AS
SELECT gid, refcat, geom
FROM catast_pol_parcelaurba;
DROP TABLE catast_pol_parcelaurba;

CREATE TABLE pre_portales
AS
SELECT gid, via, portal, geom
FROM catast_txt_portal;
DROP TABLE catast_txt_portal;
```

ANEXO V. GEOPROCESAMIENTO

Esta etapa comienza con los datos iniciales almacenados en la base de datos espacial PostGIS, y el objetivo es doble: sacar información como ayuda para la redacción de un 'Plan Municipal de Actuaciones' y preparar las capas para mostrarlas en la aplicación web. Todo con la ayuda de un software GIS de escritorio, que en este caso es el QGIS.

Algunos de los algoritmos utilizados en los geoprocesos se encuentran tanto en PostGIS como en QGIS. En este anexo se van a exponer los pasos a dar uno a uno para preparar las capas para su posterior integración en la aplicación web, pero siempre cabe la posibilidad de construir modelos en QGIS para automatizar el proceso.

Para cargar las capas que se vayan a geoprocesar en QGIS primero se accede a  para conectarse a la base de datos ya creada y desde allí se cargan todas las tablas necesarias.

Como alternativa, existe también la posibilidad de realizar los mismos pasos directamente desde el propio PostGIS mediante consultas espaciales SQL[5]. Al final lo que se pretende es tener dos modos de trabajo distintos para poder escoger cualquiera de los dos sabiendo que el resultado final no cambia. Escoger una u otra dependerá de los gustos y habilidades del usuario.

PERIODOS DE RETORNO: simplificar geometría

Es un algoritmo que solamente se encuentra entre las herramientas QGIS dentro de la 'Caja de herramientas de procesado', situado en la parte derecha de la pantalla. El nombre de la herramienta es 'Simplify geometries' (figura 66) y se encuentra dentro de los Geoalgoritmos de QGIS.

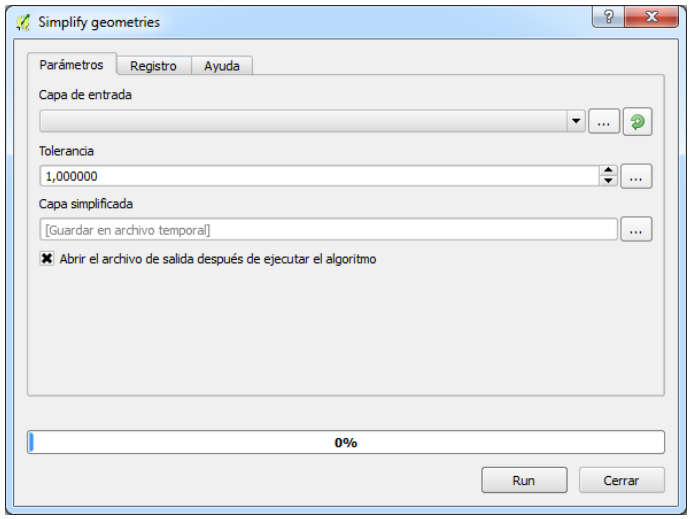


Figura 66. Herramienta 'Simplify geometries' de QGIS

Lo único que se necesita para ejecutarlo es la capa de los periodos de retorno y un valor de tolerancia para la simplificación. Este valor condicionará el resultado final y será importante definirla adecuadamente. Según las pruebas realizadas hasta el momento, con un valor de entre 5 y 8 los resultados obtenidos se ajustan a lo esperado, pero todo dependerá de la suavización que se pretenda conseguir.

Este mismo resultado se puede obtener directamente desde PostGIS mediante una consulta SQL que contenga la función *ST_Simplify (geometry, float)*, que devuelve una geometría:

```
CREATE TABLE publicar_retornos
AS
SELECT retorno, ST_Simplify (geom, 8) geom
FROM pre_periodos_retorno;
```

[5] Para todas las consultas empezar con [SET search_path=**nombre_municipio**, public] con el fin de evitar tener que escribir el esquema que contiene cada tabla.

CARRETERAS GENERALES Y CALLES: disolver, intersección, unión espacial y unión

El primer paso, disolver las líneas que componen una única calle, se lleva a cabo con la _herramienta denominada 'Dissolve' (figura 67), que necesita el nombre del campo que se pretende utilizar para disolver las líneas. En el caso de las carreteras generales ese campo es el 'DCARRETERA' y en el de las calles 'VIA'. Esta capa recién creada se utiliza tanto para calcular los puntos de intersección como para sacar la lista de calles/carreteras afectadas.

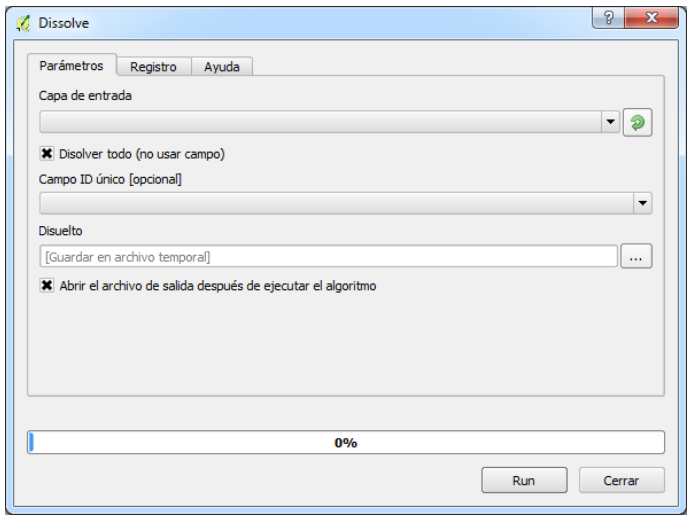


Figura 67. Herramienta 'Dissolve' de QGIS

Para calcular los puntos de intersección, el algoritmo de QGIS solamente permite trabajar con capas lineales, y como la de periodos de retorno es poligonal lo primero será convertir de poligonal a lineal con la herramienta 'Polygons to lines' (figura 68) y después aplicar el algoritmo de la herramienta 'Line intersections' (figura 69). En el primero basta con insertar la capa de los periodos de retorno que se introducirá en el segundo como capa de intersección. Los campos ID únicos serán los mismos que se han utilizado para disolver, 'VIA' y 'DCARRETERA' para cada una de las capas, y 'retorno' para la capa recién creada.

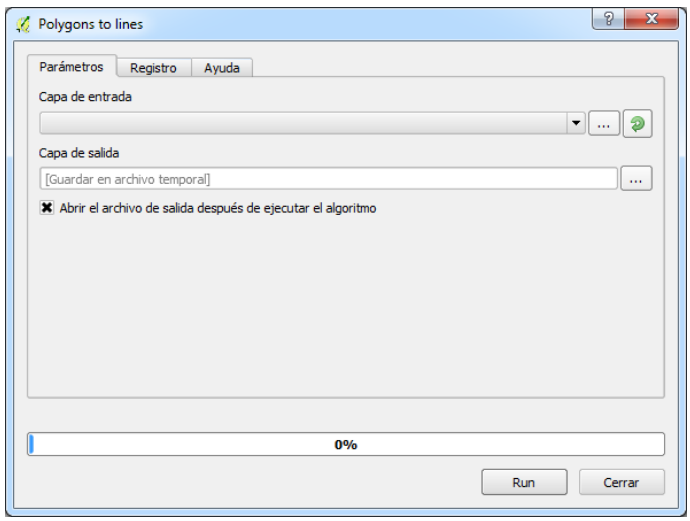


Figura 68. Herramienta 'Polygons to lines' de QGIS

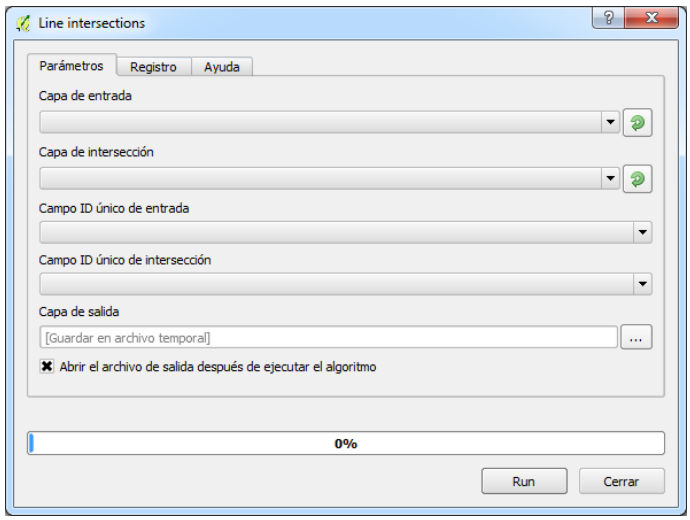


Figura 69. Herramienta 'Line intersections' de QGIS

Con estas dos acciones ya se consigue la tabla de los puntos de intersección con un campo con el nombre de la calle/carretera y el periodo de retorno de la mancha.

Para sacar la lista de las calles/carreteras afectadas se parte de nuevo de las capas originales disueltas. El método de trabajo consiste en utilizar la herramienta 'Intersection' (figura 70): la capa de entrada será la de calles/carreteras y la capa de intersección la de las manchas de inundaciones. El resultado será una nueva tabla con los elementos que quedan dentro de la mancha con los campos de ambas tablas y por lo tanto conviene eliminar las columnas que no sirven para la aplicación.

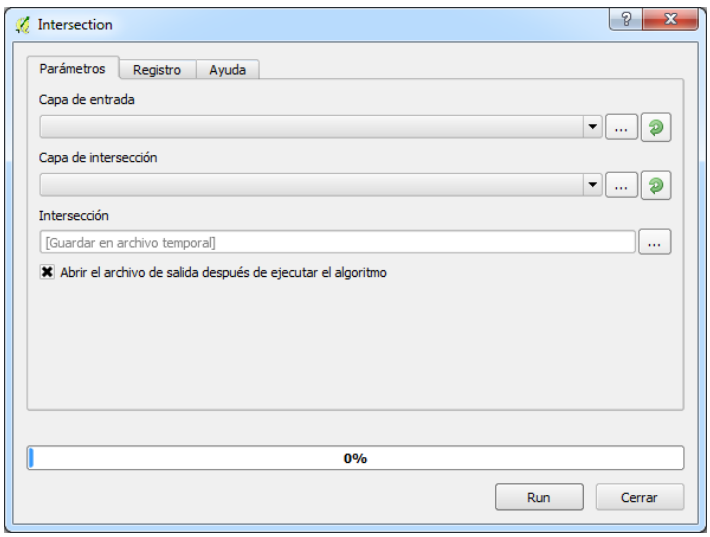


Figura 70. Herramienta 'Intersection' de QGIS

Para terminar solamente falta almacenar esta última capa en PostGIS desde el 'Administrador de BBDD'.
El mismo proceso en PostGIS se ejecutaría con las siguientes consultas SQL.
Para calcular las calles/carreteras afectadas se ha visto que son dos los pasos: disolver e intersectar. En la consulta la intersección se realiza con la función *ST_Intersection (geometry, geometry)* y será posible agruparlos todos si en *GROUP BY* se agrupa por carreteras. Para que las geometrías de todas las líneas que componen una calle/carretera se unan se utiliza la función *ST_Union*. Para que no se tengan en cuenta aquellos elementos que no intersecten con la mancha se utiliza la condición de *ST_Intersects (geometry, geometry)* que devuelve un booleano. El código para las consultas (una para cada capa) sería la siguiente:

```
CREATE TABLE publicar_carreteras
AS
SELECT p.dcarretera, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_carreteras_generales p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.dcarretera, r.retorno
ORDER BY r.retorno;

CREATE TABLE publicar_calles
AS
SELECT p.via, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_calles p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.via, r.retorno
ORDER BY r.retorno;
```

La consulta SQL para encontrar los puntos de intersección entre las manchas y las calles/carreteras es más complicada que la anterior, ya que la función *ST_Intersection* solamente devuelve la geometría de un punto cuando las dos geometrías de entrada son de tipo lineal. Para convertir las manchas de inundación, que son de tipo multi-polígono, en lineal se podría utilizar la función *ST_ExteriorRing (geometry a_polygon)*.

PUNTOS DE INTERÉS

Para determinar qué puntos de interés están afectados en cada uno de los periodos de retorno con QGIS se puede repetir el mismo proceso seguido con las calles/carreteras, pero en este caso será más fácil hacerlo con una consulta espacial SQL.

La única función necesaria es la *ST_Intersection* y escoger las columnas que se quieran mantener de cada una de las tablas:

```
CREATE TABLE publicar_puntos
AS
SELECT p.tipo, p.nombre, p.otro, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_puntos_interes p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
```

PARCELAS

El caso de las parcelas es parecido a las anteriores, ya que el objetivo es similar: conocer las porciones de polígono que se ven afectados. La diferencia con los puntos es que en este caso es posible determinar si la inundación va a llegar a toda la parcela.

Otra de las características que tienen las parcelas es la necesidad de enlazar la información de los portales con las parcelas urbanas para que estas últimas tengan campos más intuitivos que permitan identificarlas mejor.

Este trabajo se realiza en QGIS accediendo a las propiedades de la capa de las parcelas urbanas (click derecho en el nombre de la capa + Propiedades o doble click en el nombre) en la pestaña ‘Uniones’ (figura 71).

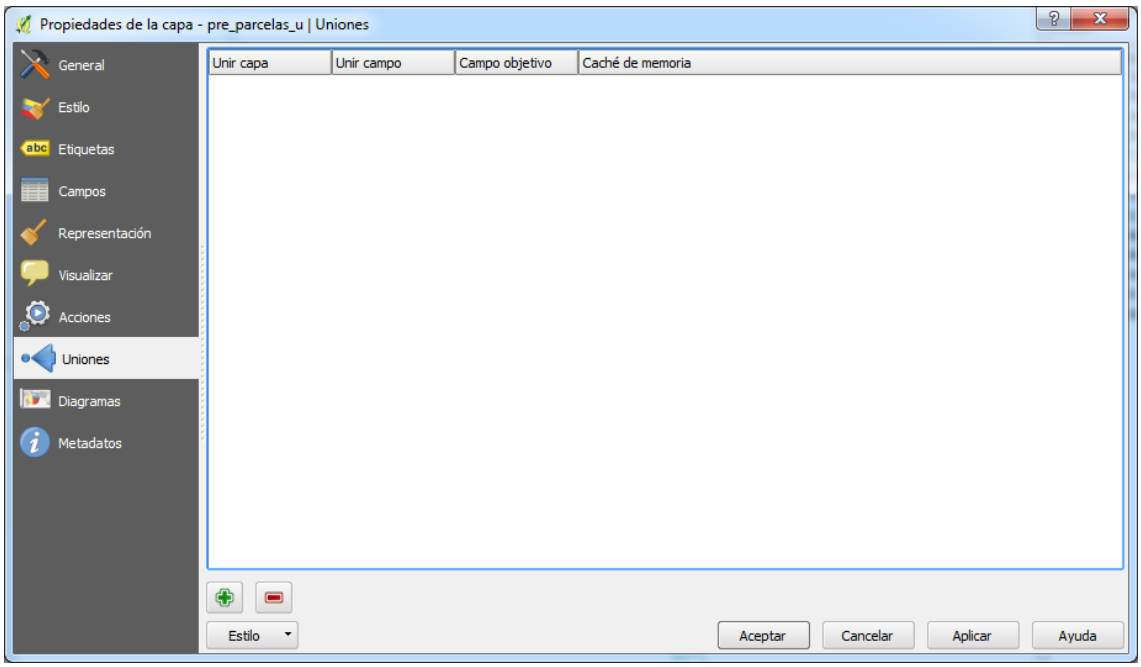


Figura 71. Propiedades de una capa en QGIS, pestaña 'Uniones'

Aquí se añade una nueva unión con  y se rellenan las opciones (figura 72) con el nombre de la capa que contiene los portales y los campos comunes para poder realizar la unión en ambos casos se llama 'refcat' (la referencia catastral). Los campos que se unen serán el 'via' y el 'portal'.

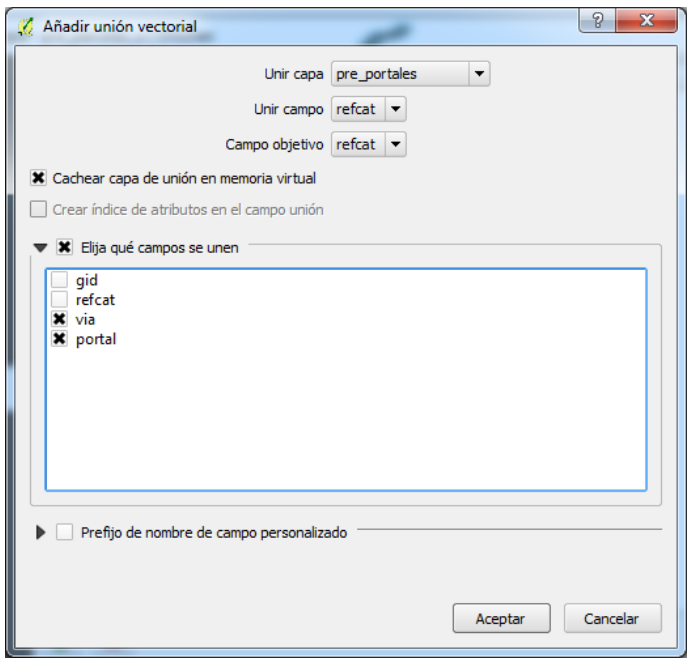


Figura 72. Opciones para la unión de tablas

Otra de las opciones es utilizar la herramienta 'Join attributes table' (figura 73). En este caso se crea una nueva capa con los campos de ambas capas de entrada y la geometría de la primera capa. De nuevo, el campo de unión para ambas tablas es el 'refcat'.

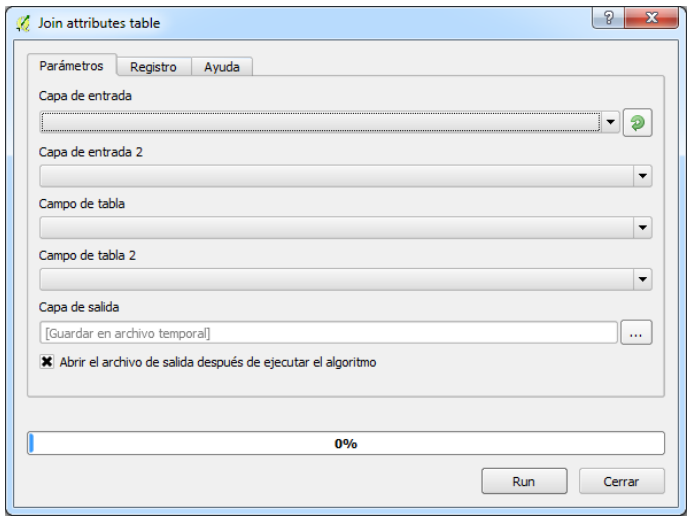


Figura 73. Herramienta 'Join attributes table' de QGIS

Una vez se ha ejecutado la unión, el siguiente paso será el geoprocesamiento, que al igual que con los elementos anteriores consiste en conocer cuáles son los que quedan dentro de la mancha y en qué periodo de retorno. Por lo tanto la herramienta es la misma: 'Intersection'.

Debido a que el cálculo de las intersecciones cuando se trata de polígonos es más costoso (las intersecciones se calculan entre líneas, y cada polígono estará formado por más de una línea) es conveniente reducir el número de parcelas antes de ejecutar el algoritmo, tanto parcelas urbanas como parcelas rústicas.

Para poder hacerlo de forma manual existe la herramienta de selección a mano alzada,  , que nos permitirá recorrer el puntero sobre las parcelas seleccionando aquellas que intersecten con el área dibujado. Para seleccionar solamente los elementos de una única capa se deberá tener seleccionada la capa.

Las parcelas seleccionadas se representarán de color amarillo (figura 74) y cuando se vea que lo seleccionado convence ya se podrá ejecutar el algoritmo, que automáticamente tiene en cuenta los elementos seleccionados. La reducción en cuanto al tiempo de ejecución es considerable.

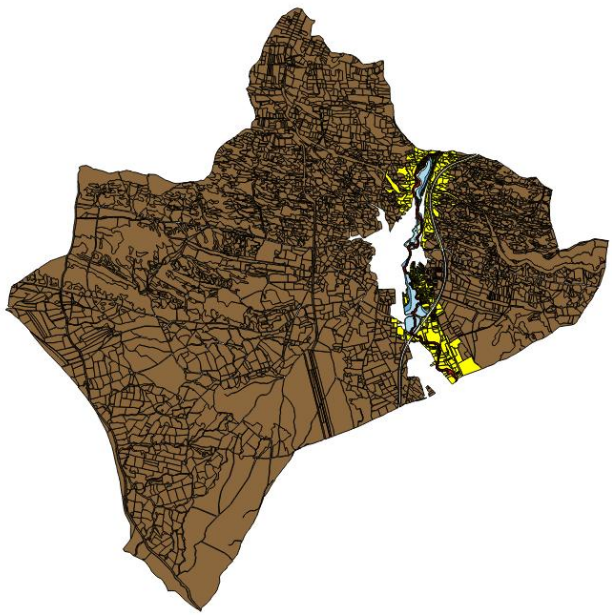


Figura 74. Parcelas rústicas de Tafalla. Seleccionadas aquellas que están cerca de las manchas de inundación

En PostGIS este mismo proceso se puede ejecutar siguiendo el mismo orden: unir tablas de portales y parcelas urbanas y después intersectar tanto el resultado del anterior como las parcelas rústicas con las manchas de inundación.

Para unir las tablas se utiliza el *LEFT JOIN* para que se le una a la tabla que primero se menciona las columnas de la segunda, pero manteniendo los registros que no se hayan podido unir con el 'refcat'. Esto ayuda a que no se eliminen aquellas parcelas que no se puedan asociar con un portal. Con esa consulta se creará una tabla temporal para que no se cree como tal pero se pueda utilizar para calcular el resultado final. El código es el siguiente:

```
CREATE TEMP TABLE portales_parcelas
AS
SELECT u.*, p.via, p.portal
FROM pre_parcelas_u u
LEFT JOIN pre_portales p ON u.refcat=p.refcat;
```

Lo siguiente sería volver a intersectar, esta vez las parcelas, con las manchas de inundación:

```
CREATE TABLE publicar_parcelas_u
AS
SELECT p.refcat, p.via, p.portal, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM portales_parcelas p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;

CREATE TABLE publicar_parcelas_r
AS
SELECT p.refcat, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_parcelas_r p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
```

CAPA DE ACTUACIÓN

Para añadir una nueva columna a la tabla de los puntos de interés de tipo texto con una longitud de 10 en PostGIS es suficiente con la siguiente consulta SQL:

```
ALTER TABLE publicar_puntos ADD actuacion character varying (10);
```

Para que sea posible introducir el identificador de la actuación que le corresponde, es necesario añadir a la tabla un 'primary key' que se crea con una columna de tipo 'serial' (conviene hacerlo con todas las tablas para evitar posibles problemas), y a continuación modificar lo que se desee con una consulta de *UPDATE*, donde se debe introducir en el campo 'actuacion' el identificador que lo relacione con una actuación específica del Plan a aquel elemento que cumpla con una condición dada.

```
ALTER TABLE publicar_puntos ADD gid serial;
ALTER TABLE publicar_puntos ADD PRIMARY KEY (gid);

UPDATE publicar_puntos
   SET actuacion='identificador_de_la_actuacion'
 WHERE condicion_que_cumpla_el_elemento_a_modificar;
```

En QGIS será posible hacerlo abriendo la tabla de la capa y siguiendo los siguientes pasos:

- Conmutar el modo de edición: 
- Columna nueva: 
- Rellenar opciones

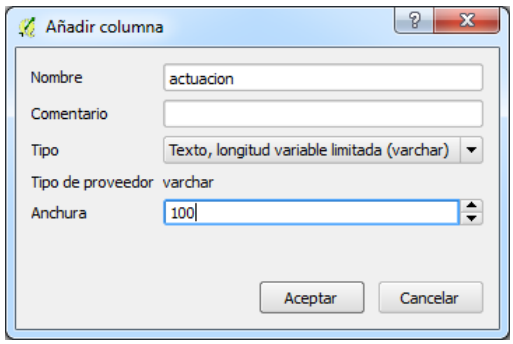


Figura 75. Añadir columna con QGIS

- Volver a conmutar el modo de edición 

MODELO CREADO PARA QGIS y CONSULTAS SQL

Para poder agilizar el trabajo en QGIS se pueden concatenar las herramientas y sus resultados mediante un modelo, que con un único click ejecutará todo la parte del geoprocesamiento hasta dejar las capas ya creadas listas para importar a la base de datos.

La única tarea antes de ejecutar el modelo es conectarse a la base de datos y añadir todas las tablas que empiezan por *pre_* y seleccionar las parcelas que estén cerca de la mancha de inundación tal y como se ha explicado en el punto anterior.

El diagrama del modelo creado en QGIS es la siguiente:

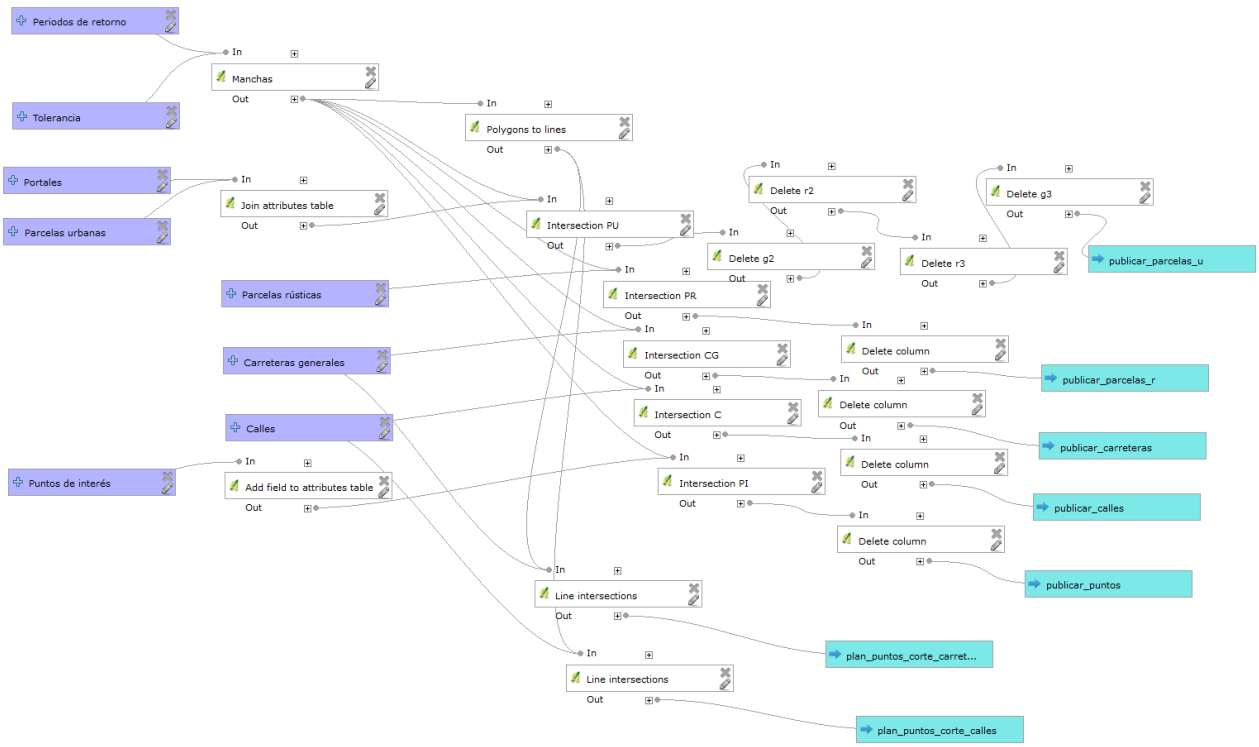


Figura 76. Diagrama del modelo de QGIS para el geoprocesamiento

Además de dejar listas las capas nuevas con el nombre adecuado para la base de datos, también carga las capas en el mapa para tener la posibilidad de supervisar el resultado, y en el caso de querer cambiarlo modificarlas antes de importarlas.

En cuanto al geoprocesamiento mediante consultas SQL, el código completo sería la siguiente:

```
SET SEARCH_PATH = nombre_municipio, public;
--########## PERIODOS DE RETORNO: seleccionar una u otra dependiendo de si hay que simplificar o
no
----CREATE TABLE publicar_retornos AS SELECT retorno, ST_Simplify (geom, 8) geom
----FROM pre_periodos_retorno;

----ALTER TABLE pre_periodos_retorno RENAME TO publicar_retornos

--######### PUNTOS
CREATE TABLE publicar_puntos
AS
SELECT p.tipo, p.nombre, p.otro, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_puntos_interes p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
--Añadir columna de 'ACTUACION' y modificarla
ALTER TABLE publicar_puntos ADD actuacion character varying (10);
ALTER TABLE publicar_puntos ADD gid serial;
ALTER TABLE publicar_puntos ADD PRIMARY KEY (gid);

UPDATE publicar_puntos
   SET actuacion='identificador_de_la_actuacion'
 WHERE condicion_que_cumpla_el_elemento_a_modificar;

--######### PARCELAS
CREATE TEMP TABLE portales_parcelas
AS
SELECT u.*, p.via, p.portal
FROM pre_parcelas_u u
LEFT JOIN pre_portales p ON u.refcat=p.refcat;

CREATE TABLE publicar_parcelas_u
AS
SELECT p.refcat, p.via, p.portal, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM portales_parcelas p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
----

CREATE TABLE publicar_parcelas_r
AS
SELECT p.refcat, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_parcelas_r p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;

--######### CARRETERAS Y CALLES
CREATE TABLE publicar_carreteras
AS
SELECT p.dcarretera, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_carreteras_generales p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.dcarretera, r.retorno
ORDER BY r.retorno;
-----
CREATE TABLE publicar_calles
AS
SELECT p.via, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_calles p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.via, r.retorno
ORDER BY r.retorno;
```

RECOMENDACIÓN

Lo más cómodo va a ser trabajar directamente desde el PostGIS y ejecutar una a una todas las consultas SQL para no tener que estar exportando e importando y utilizar el modelo de QGIS para aquellos casos en los que la consulta no se pueda realizar, por ejemplo porque no hay suficiente espacio en el disco (demasiadas líneas o polígonos para analizar):

ERROR: could not write block 227220 of temporary file: No space left on device

Para estos casos utilizar el modelo completo y calcular cada una de las capas nuevas puede resultar excesivo, pero siempre se puede escoger añadir al mapa solamente la capa que falta en la base de datos e importar solamente ésta (figura 77). Otra posibilidad es ejecutar solamente las herramientas necesarias correspondientes expuestas anteriormente.

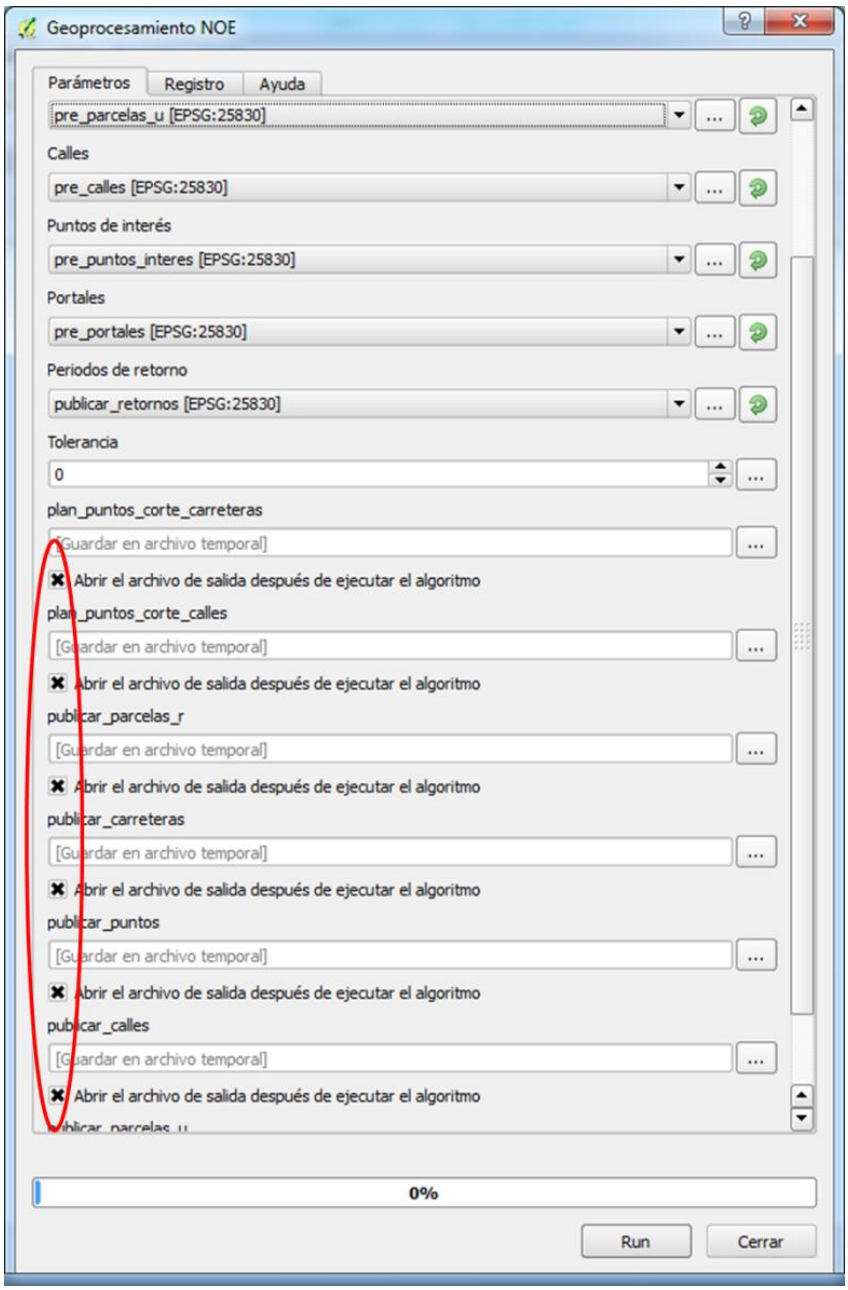


Figura 77. Interfaz para ejecutar el modelo creado en QGIS

En el caso de querer trabajar con vistas, el código SQL completo (pre-procesamiento + geoprocesamiento) es la siguiente (hay que tener en cuenta que con QGIS no se podrán crear vistas):

```
SET SEARCH_PATH=nombre_municipio, public;

-----PRE-PROCESAMIENTO DE LOS DATOS
CREATE TABLE pre_calles
AS
SELECT gid, via, geom
FROM catast_lin_calleeje;
DROP TABLE catast_lin_calleeje;
----
CREATE TABLE pre_contorno
AS
SELECT gid, municipio, geom
FROM catast_pol_municipio;
DROP TABLE catast_pol_municipio;
----
CREATE TABLE pre_parcelas_r
AS
SELECT gid, refcat, geom
FROM catast_pol_parcelarusti;
DROP TABLE catast_pol_parcelarusti;
----
CREATE TABLE pre_parcelas_u
AS
SELECT gid, refcat, geom
FROM catast_pol_parcelaurba;
DROP TABLE catast_pol_parcelaurba;
----
CREATE TABLE pre_portales
AS
SELECT gid, refcat, via, portal, geom
FROM catast_txt_portal;
DROP TABLE catast_txt_portal;
----
CREATE VIEW pre_carreteras_generales
AS
SELECT c.gid, c.dcarretera, c.carretera, ST_Intersection(c.geom, m.geom) geom
FROM infrae_lin_ctraeje c, pre_contorno m
WHERE ST_Intersects(c.geom, m.geom);
----
CREATE VIEW pre_periodos_retorno
AS
SELECT r.gid, retorno, ST_Intersection(r.geom, m.geom) geom
FROM hidrog_pol_periodoret r, pre_contorno m
WHERE ST_Intersects(r.geom, m.geom);
----
CREATE VIEW pre_puntos_interes
AS
SELECT p.id, tipo, nombre, otro, ST_Intersection(p.geom, m.geom) geom
FROM puntos_interes p, pre_contorno m
WHERE ST_Intersects(p.geom, m.geom);

----------GEOPROCESAMIENTO
--########## PERIODOS DE RETORNO: seleccionar una u otra dependiendo de si hay que simplificar o
no
----CREATE VIEW publicar_retornos AS SELECT retorno, ST_Simplify (geom, 8) geom
----FROM pre_periodos_retorno;

----ALTER VIEW pre_periodos_retorno RENAME TO publicar_retornos

--######### PUNTOS
CREATE VIEW publicar_puntos
AS
SELECT p.tipo, p.nombre, p.otro, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_puntos_interes p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
```

```
--Añadir columna de 'ACTUACION' y modificarla
ALTER VIEW publicar_puntos ADD actuacion character varying (10);
ALTER VIEW publicar_puntos ADD gid serial;
ALTER TABLE publicar_puntos ADD PRIMARY KEY (gid);

UPDATE publicar_puntos
   SET actuacion='identificador_de_la_actuacion'
 WHERE condicion_que_cumpla_el_elemento_a_modificar;

--######### PARCELAS
CREATE TABLE pre_portales_parcelas_u
AS
SELECT u.*, p.via, p.portal
FROM pre_parcelas_u u
LEFT JOIN pre_portales p ON u.refcat=p.refcat;
DROP TABLE pre_portales;
DROP TABLE pre_parcelas_u;

CREATE VIEW publicar_parcelas_u
AS
SELECT p.refcat, p.via, p.portal, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_portales_parcelas_u p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;
----

CREATE VIEW publicar_parcelas_r
AS
SELECT p.refcat, r.retorno, ST_Intersection(p.geom, r.geom) geom
FROM pre_parcelas_r p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
ORDER BY r.retorno;

--######### CARRETERAS Y CALLES
CREATE VIEW publicar_carreteras
AS
SELECT p.dcarretera, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_carreteras_generales p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.dcarretera, r.retorno
ORDER BY r.retorno;
-----
CREATE VIEW publicar_calles
AS
SELECT p.via, r.retorno, ST_Union(ST_Intersection(p.geom, r.geom)) geom
FROM pre_calles p, publicar_retornos r
WHERE ST_Intersects(p.geom, r.geom)
GROUP BY p.via, r.retorno
ORDER BY r.retorno;
```

ANEXO VI. PUBLICACIÓN DE CAPAS

Se dice que una capa está publicada cuando existe un servicio web estandarizado por la OGC que permite a otras herramientas GIS acceder a ellas a través de la web para poder utilizarlas después. En este caso el servidor para compartir y editar datos geoespaciales es el GeoServer, que es un servidor de código abierto.

Son tres los pasos a dar para publicar capas en GeoServer: crear un espacio de trabajo, crear un almacén de datos y publicar capas. Además también es posible añadir estilos a las capas y pre-visualizar en distintos formatos lo que se está sirviendo. Se accede a cada una de las opciones a través del apartado 'Datos' que se encuentra en la parte izquierda de la interfaz de la aplicación web (figura 79) que proporciona GeoServer para su gestión.

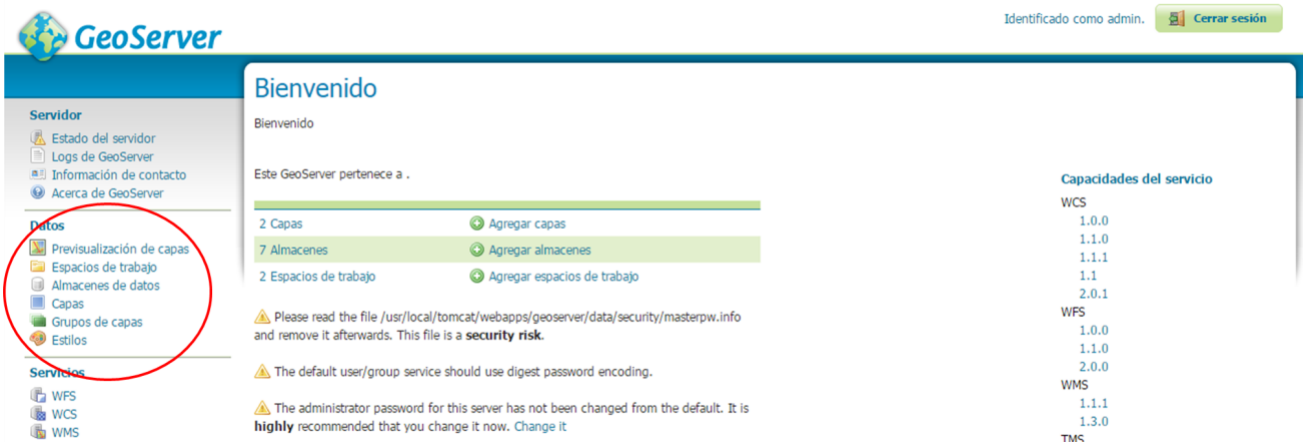


Figura 78. Interfaz de administración web de GeoServer

PASO 1: CREAR UN ESPACIO DE TRABAJO

Los espacios de trabajo se crean pulsando 'Agregar un nuevo espacio de trabajo':

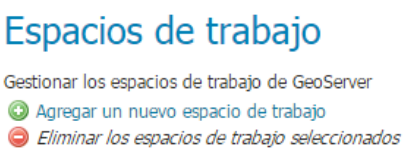


Figura 79. Crear un nuevo espacio de trabajo

Se rellena con un nombre que describe el proyecto y una URI y se pulsa 'Enviar'. Conviene volver al espacio de trabajo para editar y habilitar los servicios WMS y WFS:

Figura 80. Edición del espacio de trabajo

PASO 2: CREAR UN ALMACÉN DE DATOS

Lo próximo es asignar a ese espacio de trabajo recién creado un almacén de datos que determine la fuente de los datos, que en este caso se trata del PostGIS que se ha ido construyendo en los procesos anteriores.

Cuando se crea un almacén de datos de tipo PostGIS (figura 81) solamente es posible crear uno por esquema, y esto significa que habrá uno por municipio.

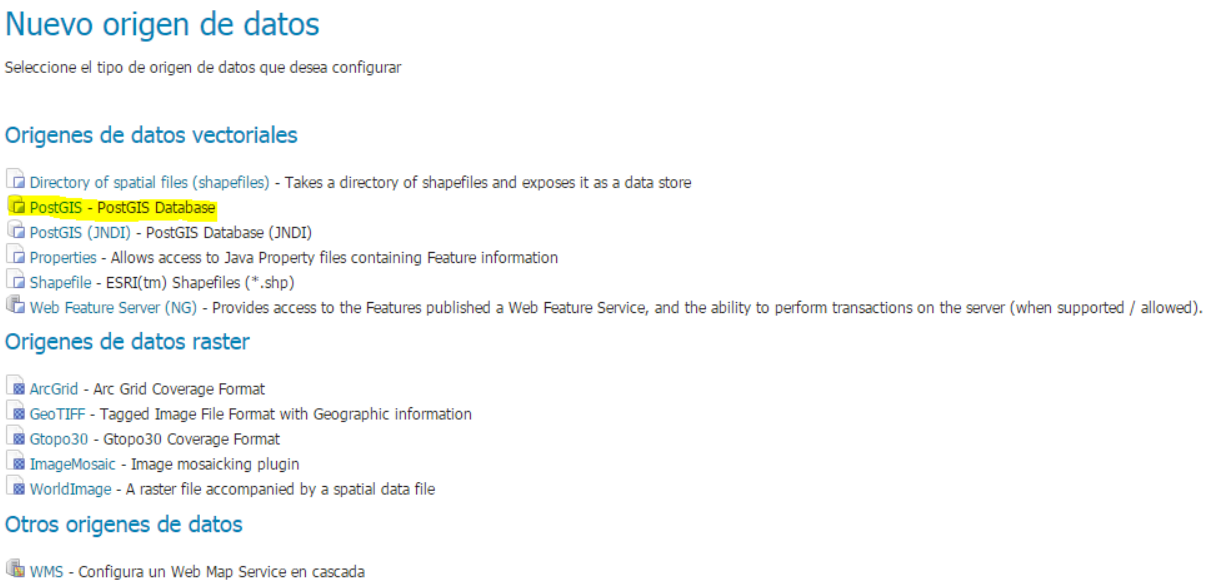


Figura 81. Orígenes de datos para crear un almacén de datos en GeoServer

Dependiendo de los nombres empleados en la base de datos se tendrán que rellenar las opciones (figura 82) adecuadamente. Las opciones a rellanar son el nombre, habilitar y poner los datos de la base de datos al que se conectará: host, puerto, nombre de la base de datos, nombre de esquema y usuario y contraseña. Todo dependerá de lo hecho en procesos anteriores.

Figura 82. Opciones para conectar GeoServer con un esquema de PostGIS

PASO 3: PUBLICAR CAPAS

Primero se agrega un nuevo recurso:

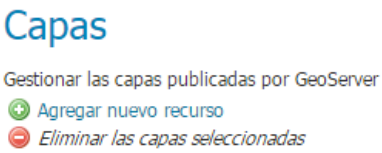


Figura 83. Agregar nuevo recurso

Después, tras elegir el almacén de datos donde se encuentra la capa y clicar la opción 'Publicación' de la capa en cuestión, se rellenan las opciones (figura 84) de la pestaña 'Datos' con el nombre, SRS y los encuadres, que se pueden calcular automáticamente desde los datos o introducirlos a mano.

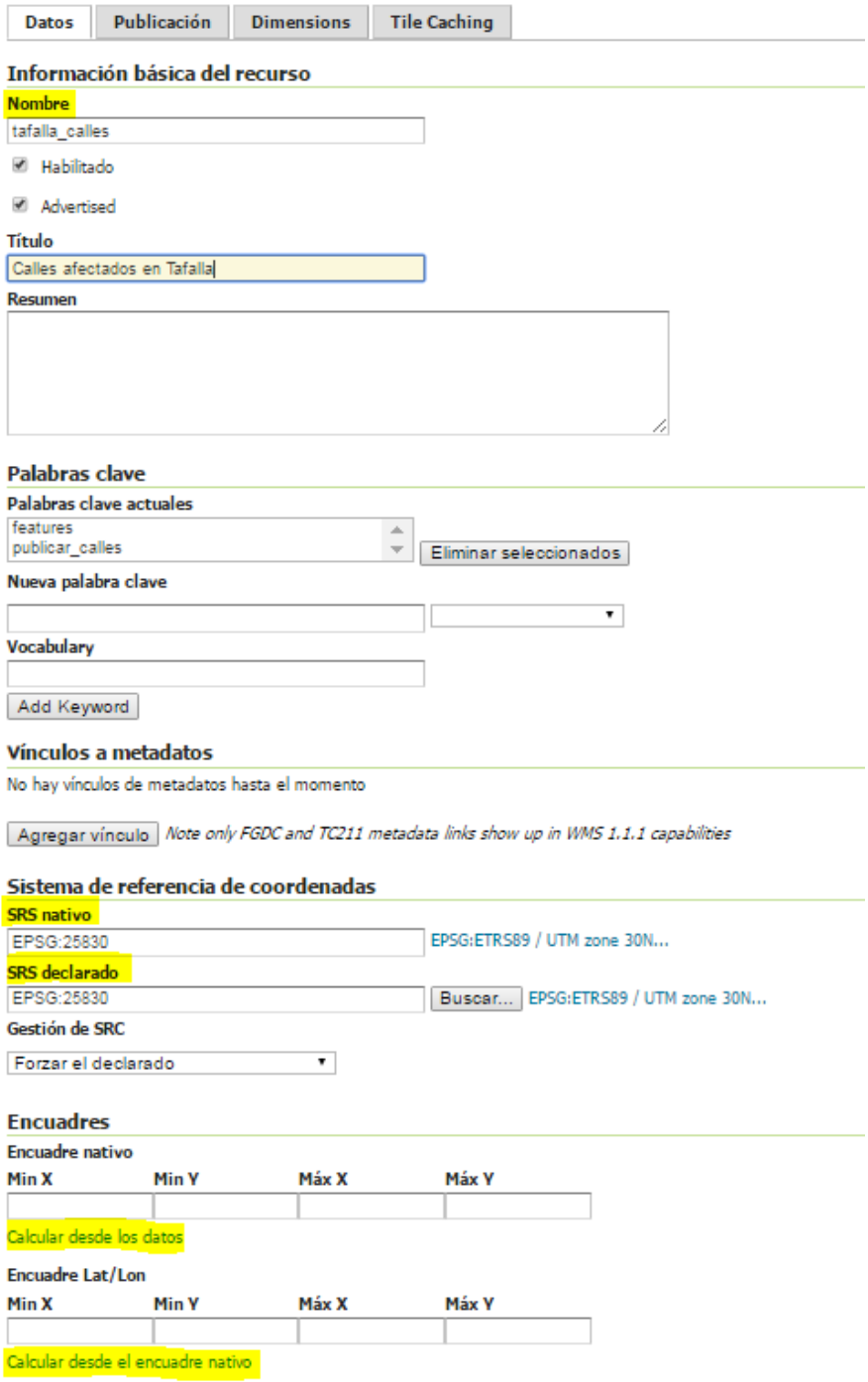


Figura 84. Opciones para la publicación de los datos

En la pestaña de 'Publicación' conviene asegurarse de que el estilo por defecto sea del tipo que le corresponde a la capa: puno, línea o polígono. De este modo se asegura que lo va a representar correctamente cuando se ejecute una petición WMS.

PRE-VISUALIZACIÓN DE CAPAS

Para comprobar que lo que se ha publicado es correcto existe la posibilidad de visualizarlo en distintos formatos. Lo habitual es visualizar el WMS con OpenLayers, que permite supervisar la capa tal y como se representaría en una aplicación web.

En este apartado también es interesante conocer tanto el nombre de la aplicación como la URL del servicio web. El nombre de la capa aparecerá en el campo 'Nombre' de la tabla del apartado 'Previsualización de capas' (figura 85), y se compone del nombre del espacio de trabajo, dos puntos y nombre de la capa.

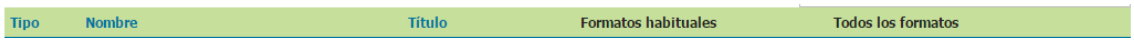


Figura 85. Campos de la tabla de 'Previsualización de capas'

La URL del servicio se podrá consultar abriendo la capa en formato OpenLayers y la parte hasta la '?' será la URL del servicio web (figura 86) que es útil para poder hacer una llamada a la capa desde la API a la hora de integrarla en la aplicación web. Normalmente se compone con:

https://servidor/geoserver/espacio_trabajo/wms_wfs

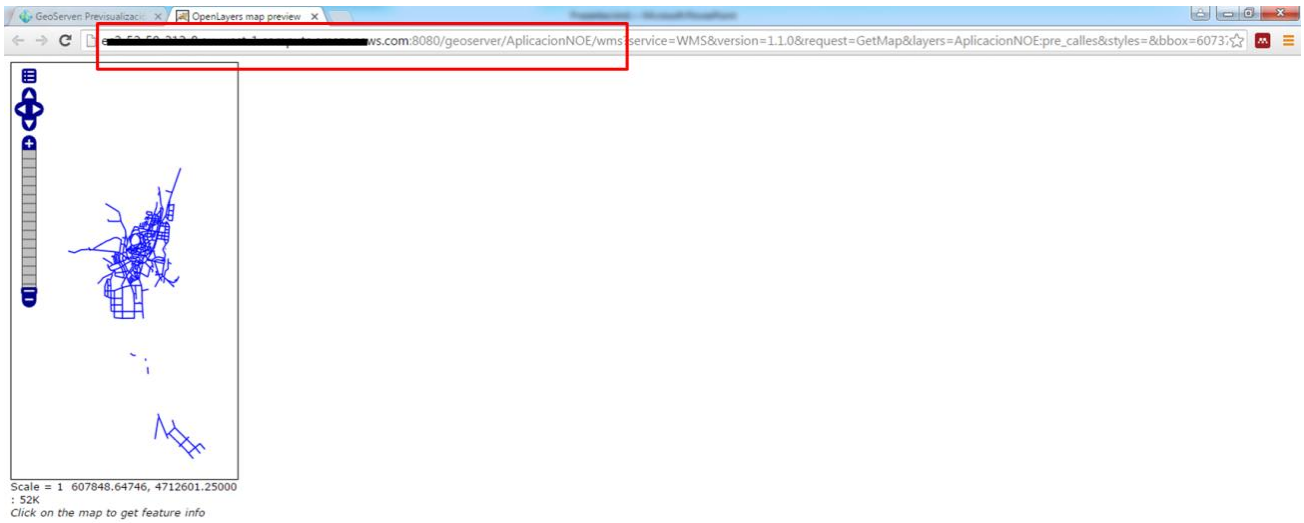


Figura 86. URL del servicio web que publica la capa

ANEXO VII. CÓDIGO PARA LA VISUALIZACIÓN DE LOS DATOS GEOGRÁFICOS EN UNA APLICACIÓN WEB

El siguiente código ejecutable hace referencia a un HTML cuya integración en la aplicación web NOE sea fácil y sencilla, es decir, en las siguientes líneas no se describe la aplicación en su totalidad, si no que se ha diseñado un visor junto a un pequeño formulario que se parezca a la aplicación web NOE para que, cambiando unas pocas referencias a objetos, sea posible agregar las capas creadas junto a algunas características (pop-ups, comunicación visor/actuaciones…) a la aplicación final.

La idea es mostrar partes del código ejecutable de los tres archivos (HTML, JavaScript y CSS) que se consideren de relevancia o que necesiten algún tipo de explicación para su compresión.

HTML

La estructura de la página web está formada por un mapa y un formulario donde se distinguen:

- Un seleccionador del periodo de retorno con un botón para aplicarlo. En el caso de integrar este código ejecutable en la aplicación web NOE, está acción no se ejecutaría pulsando un botón, si no que cuando en la base de datos se identifique que ha habido un cambio de Nivel de Alerta.
- Actuaciones de ejemplo con *radio button* para cambiar el estado. En una integración de la aplicación original, estas actuaciones se cargarían junto al mapa cuando se note que ha habido un cambio en el Nivel de Alerta. Estas actuaciones están relacionadas con los elementos del visor a través de un identificador único (el campo 'actuacion' de la capa puntos).
- Checkbox para activar la edición de una capa vacía y que así el usuario tenga la posibilidad de proponer una modificación de las capas originales. Al activarlo se añaden al mapa una capa vacía el control de edición. Con las opciones de 'Cambiar forma' y 'Trasladar' se podrán modificar los elementos creados pudiendo filtrar por puntos, líneas y polígonos. El botón de ´Limpiar' elimina todos los elementos de la capa y 'Enviar propuesta de modificación' guarda la geometría de la capa en la base de datos espacial para que el desarrollador lo pueda consultar y cambiar las capas originales que se están publicando.

JavaScript

Siguiendo el orden de lo redactado en el trabajo, lo primero es crear un mapa y añadirle opciones como número máximo de zooms o la extensión máxima para poder delimitar la zona que abarca el visor. Posteriormente se añade una capa base de tipo ortofoto y se le añaden capas vacías, tantas como capas se van a cargar después. En esta primera parte también se declaran los estilos por defecto de las distintas capas:

```
style = new OpenLayers.StyleMap({
        "default": new OpenLayers.Style({
            strokeColor: "#417dd4",
            strokeOpacity: 0.6,
            fillColor: "#91c2d9",
            fillOpacity: 0.4
        })
    });
```

Con estas primeras líneas se define el aspecto de la aplicación nada más abrirlo. Todas las funciones que se describen a partir de ahora se ponen en marcha solamente cuando se pulsa algún botón del formulario descrito anteriormente.

La primera función se activa al escoger un periodo de retorno en el formulario[6]. En ese instante se cargan las capas que se han creado en los pasos anteriores. Para ello se cargan las capas servidas por el servidor de mapas en WFS y se filtran dependiendo del periodo de retorno que se ha escogido en el formulario. También se le asigna un estilo por defecto (que será alguno de los estilos definidos al principio).

```
calles = new OpenLayers.Layer.Vector("Calles", {
        protocol: new OpenLayers.Protocol.WFS({
            version: "1.1.0",
            url: "http://ec2-52-210-134-105.eu-west-
1.compute.amazonaws.com:8080/geoserver/AplicacionNOE/wfs",
            featureType: "publicar_calles",
            featureNS: "http://ec2-52-209-15-168.eu-west-1.compute.amazonaws.com:8080/",
            geometryName: "geom"
        }),
        styleMap: styleLIN,
        strategies: [new OpenLayers.Strategy.Fixed()],
        filter: new OpenLayers.Filter.Logical({
            type: OpenLayers.Filter.Logical.AND,
            filters: [
                    new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.GREATER_THAN,
                        property: "retorno",
                        value: perRetorno
                    }),
                    new OpenLayers.Filter.Comparison({
                        type: OpenLayers.Filter.Comparison.LESS_THAN,
                        property: "retorno",
                        value: perRetorno2
                    })
            ]
        })
    });

    mapa.addLayer(calles);
```

Después, a cada una de estas capas se les registra un evento para que al seleccionarlas aparezcan ventanas emergentes que muestren cierta información sobre la capa y que desaparezcan al deseleccionarlas. En el caso de los puntos de interés también se añade una función que cambia el color de la actuación con la que está relacionada en el formulario:

```
//Gestor de evento para activar selección
    puntos.events.register("featureselected", null, function(event){
        var layer = event.feature.layer;
        if (event.feature.attributes.actuacion==null) {
            event.feature.style = {
            pointRadius: 20,
            fillOpacity: 1,
            externalGraphic: './iconos/info.png'
            };
        };
        layer.drawFeature(event.feature);
        if (event.feature.attributes.otro) {
            var content =  "<div><strong>"+event.feature.attributes.tipo +":</strong> <br/>" +
event.feature.attributes.nombre + " (" +event.feature.attributes.otro +")"+"</div>";
        } else {
            var content =  "<div><strong>"+event.feature.attributes.tipo +":</strong> <br/>" +
event.feature.attributes.nombre + "</div>";
        }
        var popup = new OpenLayers.Popup.FramedCloud(
            event.feature.id+"_popup",
            event.feature.geometry.getBounds().getCenterLonLat(),
            new OpenLayers.Size(250, 100),
            content,
            null,
            false,
            null);
        event.feature.popup = popup;
```

[6] En realidad, si estuviese integrado en la aplicación web NOE, está función se debería iniciar cuando se cambie de nivel de emergencia. Esto quiere decir que mediante métodos AJAX se tendría que extraer el valor del nivel de emergencia que se almacena en la base de datos que recoge la información de los sensores SAIH, y dependiendo de este valor mostrar los datos adecuados

```
        mapa.addPopup(popup);

        if(event.feature.attributes.actuacion){
            var act = event.feature.attributes.actuacion;
            var element = document.getElementById(act);
            element.style.backgroundColor="#1803ff";
        };
    });
    //Gestor de evento para desactivar selección
    puntos.events.register("featureunselected", null, function(event){
        var layer = event.feature.layer;
        if (event.feature.attributes.actuacion==null) {
            event.feature.style = null;
        };
        event.feature.renderIntent = null;
        layer.drawFeature(event.feature);
        mapa.removePopup(event.feature.popup);

        if(event.feature.attributes.actuacion){
            var act = event.feature.attributes.actuacion;
            var element = document.getElementById(act);
            element.style.backgroundColor="#3498DB";
        };
    });
    // Añadir control de selección a la capa vectorial para activar eventos.
    var selectControl = new OpenLayers.Control.SelectFeature(puntos);
    mapa.addControl(selectControl);
    selectControl.activate();
```

Otra de las funciones que realiza la aplicación tiene que ver con el estado de las actuaciones: al cambiar de estado en el formulario el icono del mapa que representa ese punto también tiene que cambiar de rojo a naranja y de naranja a verde. Está función se inicia al pulsar una de las opciones que ofrece el *radio button* en el formulario.

```
function progress0PREChanged(){
    var feature = puntos.getFeaturesByAttribute('actuacion', "act0");
    puntos.drawFeature(feature[0], {pointRadius: 20, fillOpacity: 1, externalGraphic:
        './iconos/buzPRE.png' });
    feature[0].style={pointRadius: 20, fillOpacity: 1, externalGraphic: './iconos/buzPRE.png' };
}
```

Para finalizar quedan las funcionalidades relacionadas con la edición de una capa vacía para que el usuario tenga la opción de realizar una propuesta de mejora. Para ello lo primero será seleccionar la opción en el formulario y en ese instante se añade una capa vacía y los controles de edición y modificación. A partir de ese momento se podrán dibujar puntos, líneas y recintos en el mapa para posteriormente cambiar las propiedades (trasladar y cambiar forma), filtrar por tipo de geometría o limpiar lo dibujado.

```
function activateEdition(){
    ////////////////////////////////////// MODIFICACIONES
    vectorLayer = new OpenLayers.Layer.Vector("Capa vectorial para mejorar Plan");
    mapa.addLayer(vectorLayer);
    //Controles para editar las características
    var editingToolbarControl = new OpenLayers.Control.EditingToolbar(vectorLayer);
    //CUIDADO: Se crea como variable global para que los eventos puedan modificarlo
    modifyControl = new OpenLayers.Control.ModifyFeature(vectorLayer);
    mapa.addControls([editingToolbarControl, modifyControl]);

}


//Modificar la propiedad del modo
function changeMode() {
    var reshape = document.getElementById("Reshape").checked;
    var drag = document.getElementById("Drag").checked;
    var mode = null;
    if(reshape) {
        mode |= OpenLayers.Control.ModifyFeature.RESHAPE;
    }
    if(drag) {
        mode |= OpenLayers.Control.ModifyFeature.DRAG;
    }
    modifyControl.deactivate();
    modifyControl.mode = mode;
    modifyControl.activate();
```

```
}
//Modificar la propiedad 'geometryType', para ello hay que redefinir el control.
function changeFilter() {
    var value = document.getElementById('sFilter').value;
    modifyControl.deactivate();
    mapa.removeControl(modifyControl);
    modifyControl.destroy();
    var geometryTypes = null;
    if(value=="POINT") {
        geometryTypes = ["OpenLayers.Geometry.Point"];
    } else if(value=="PATH") {
        geometryTypes = ["OpenLayers.Geometry.LineString"];
    } else if(value=="POLYGON") {
        geometryTypes = ["OpenLayers.Geometry.Polygon"];
    }
    modifyControl = new OpenLayers.Control.ModifyFeature(vectorLayer, {
        geometryTypes: geometryTypes
    });
    mapa.addControl(modifyControl);
    modifyControl.activate();
    changeMode();
}

function clearLayer() {
    vectorLayer.removeAllFeatures();
}
```

La última función consistiría en guardar en la base de datos espacial la capa que ha sido modificada por el usuario. Esta función se llevaría a cabo con una transacción del WFS-T que GeoServer lo interpretaría para almacenar la geometría que ha sido enviada en formato XML. Para ello sería necesario publicar primero una capa vacía, es decir, tener una tabla vacía en PostGIS que se vaya a publicar, realizar modificaciones en la aplicación y que mediante técnicas AJAX se le transmita a GeoServer con un XML la nueva geometría de la capa para que este lo almacene en PostGIS por medio del servicio WFS-T.